НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Інформаційна система удосконалення сприйняття інформації технології доповненої реальності»

Виконала: студентка *IV* курсу, групи *ІС-63*

Швець Дар'я Юріївна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст. вик. Проскура Світлана Леонідівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *ст. вик. Проскура Світлана Леонідівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц., к.т.н. доц. Лісовиченко Олег Іванович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність –122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючісистеми та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім`я, прізвище)

“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Швець Дар’ї Юріївни
(прізвище, ім`я, по батькові)

1. Тема проєкту«Інформаційна система удосконалення сприйняття інформації технології доповненої реальності »
керівник проєкту*Проскура Світлана Леонідівна, ст. викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня*2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня*2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використання

2. Схема бази даних

3. Схема структурна класів програмного забезпечення

4. Схема структурна послідовності

5. Архітектура програмного забезпечення

6. Креслення електронних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання*«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*14.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*16.04.2020*	
3.	*Постановка та формалізація задачі*	*18.04.2020*	
4.	*Розробка інформаційного забезпечення*	*22.04.2020*	
5.	*Алгоритмізація задачі*	*26.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*27.04.2020*	
7.	*Розробка програмного забезпечення*	*29.04.2020*	
8.	*Налагодження програми*	*09.05.2020*	
9.	*Виконання графічних документів*	*10.05.2020*	
10.	*Оформлення пояснювальної записки*	*11.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Дар'я ШВЕЦЬ

Керівник Світлана ПРОСКУРА

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6330.00.000 ПЗ	Пояснювальна записка	115	
3	А4	ДП 6330.01.000 ТЗ	Технічне завдання	10	
4	А3	ДП 6330.02.000 ССВ	Схема структурна варіантів	1	
			використання		
5	А3	ДП 6330.03.000 СБД	Схема бази даних	1	
6	А3	ДП 6330.04.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
7	А3	ДП 6330.05.000 ССП	Схема структурна послідовності	1	
8	А3	ДП 6330.06.000 АПЗ	Архітектура програмного забезпечення	1	
9	А3	ДП 6330.07.000 ЕК	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП6330 00.000.00		
Розробн.		Швець Д.Ю.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		ПроскураС.Л.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63	
Н/контр.		Проскура С.Л.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Інформаційна система удосконалення сприйняття інформації технології доповненої реальності

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з шести розділів, містить 8 рисунків, 28 таблиць, 1 додаток, 13 джерел.

Дипломний проєкт присвячений розробці інформаційної системи удосконалення сприйняття інформації технології доповненої реальності. Цілями створення системи є полегшення створення 3D-моделей, оптимізація етапу накладання 3D-моделей на об’єкти та полегшення управління доповненою інформацією для усіх товарів. Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

– створення 3D-моделей;
– аналіз реального світу через камеру смартфона;
– накладання моделі на певний об’єкт з навколишнього середовища;
– вивід товару з доповненою реальністю на ринок.

У розділі інформаційного забезпечення описано вхідні та вихідні дані системи, описано базу даних та надано детальний опис кожної з таблиць.

Розділ математичного забезпечення присвячений опису математичної задачі, яка наявна в системі, обґрунтовано вибір алгоритму та описано його.

Програмне забезпечення, вимоги до програмного забезпечення, засоби розробки, архітектура описані у четвертому розділі.

У технологічному розділі описано керівництво користувача та проведено випробування програмного продукту.

КЛЮЧОВІ СЛОВА: ДОПОВНЕНА РЕАЛЬНІСТЬ, 3D-МОДЕЛІ, КОМП’ЮТЕРНИЙ ЗІР, SIFT.

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of six sections, contains 8 images, 28 tables, 1 appendices, 13 sources.

The diploma project is devoted to the development of an information system to improve the perception of information by augmented reality technology. The goals of the system are to facilitate the creation of 3D models, to optimize the stage of superimposing 3D models on objects and to facilitate the management of supplemented information for all products. To achieve these goals it is necessary to solve the following tasks:

– creation of 3D-models;
– analysis of the real world through a smartphone camera;
– applying a model to a specific object from the environment;
– withdrawal of goods with augmented reality on the market

The information support section describes the input and output data of the system, the database and provides a detailed description of each of the tables.

The section of mathematical software is devoted to the description of the mathematical problem, which is available in system, the choice of algorithm is substantiated and described.

Software, software requirements, development tools, architecture are described in the fourth section.

The technological section describes the user manual and tests of the software product.

KEY WORDS: AUGMENTED REALITY, 3D MODELS, COMPUTER VISION, SIFT.

ЗМІСТ

ВСТУП

В останній час активно розвивається така сфера IT як доповнена та віртуальна реальності. Відмінність між цими двома поняттями у наступному: віртуальна реальність – це вигаданий, технічно створений світ, який передається людині, у доповненій реальності об’єкти накладаються на реальне оточення. Отже, доповнена реальність проєктує будь-яку інформацію на екран пристрою. Дана технологія може бути реалізована за допомогою мобільних додатків, окулярів доповненої реальності, проекційних пристроїв.

Для реалізації доповненої реальності у телефоні необхідно розуміти, що додаток повинен через камеру зчитувати навколишнє середовище, визначаючи розміри та місцезнаходження об’єктів, а також своє положення у цьому просторі. Далі сервіс розміщує віртуальний об’єкт відносно усіх об’єктів у реальному світі, використовуючи координати речей задля гармонійного знаходження його на екрані смартфона.

Дипломний проєкт присвяченийрозробці комплексу задач:

- створення 3D-моделей;
- аналіз реального світу через камеру смартфона;
- накладання моделі на певний об’єкт з навколишнього середовища;
- вивід товару з доповненою реальністю на ринок.

Практичне значення одержаних результатів. Розроблено алгоритм розпізнавання предметів на зображенні та побудови доповненої реальності.

Публікації. Результати роботи були опублікованів тезах доповідей на науково-технічній конференції [1].

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовища

Об’єктом, що автоматизується, в даному дослідженні, є процеси впровадження доповненої реальності (AR) та сприйняття її користувачами. Доповнена реальність – це представлення реального світу з покращенням його через інформаційні технології. Даними для доповненої реальності можуть бути зображення, відео, звуки, та багато іншого. Дана технологія може застосовуватися у всіх сферах життя людини, саме тому вона набуває все більшої популярності. Вже зараз її можна зустріти у таких областях, як архітектура, будівництво, наука, медицина, ігри, навігація і т.д.

Проаналізувавши ринок доповненої реальності, було виявлено, що на даний момент немає єдиної зручної системи для використання цієї технології саме для користувача, а компаніям доводиться витрачати багато грошей на її впровадження на підприємстві, навіть якщо їм потрібні невеликі проєкти. Виявляється, що часто компанії не йдуть у цю сферу через те, що витрати будуть вищі, ніж прибуток від її впровадження. Також, для користувачів є не дуже зручним завантаження різних додатків для різних компаній чи цілей. Отже, постає питання, яким чином зменшити витрати на впровадження доповненої реальності на підприємстві, а також удосконалити використання цієї технології.

Задача роботи дипломного проєкту і полягає у тому, щоб дозволити компаніям спрощено використовувати AR для своїх продуктів задля збільшення інтересу у клієнтів, і відповідно, збільшення користування певними товарами; а також просто людям мати можливість передавати, отримувати та використовувати інформацію засобами AR.

Для вирішення задачі впровадження та сприйняття інформації засобами доповненої реальності необхідно виконати такі роботи:

– формування методів створення 3D моделей;

- розрахунок коректного накладання 3D моделей на об’єкти;
- формування методу ідентифікації (сканування) товару за його зовнішнім виглядом;
- вивід моделі на зображення камери.

Найскладнішим завданням при побудові системи є саме накладання 3D моделей на реальний світ, адже програма повинна розуміти, де і яким чином найкраще відобразити цей об’єкт.

1.1.1. Опис процесу діяльності

Система створення і сприйняття доповненої реальності дає можливість користувачам дивитись каталог 3D-моделей, створювати нові моделі, накладати моделі на будь-які об’єкти, сканувати предмети для відображення доповненої реальності.

При скануванні об’єктів, у додатку запускається камера, яка зчитує предмет за багатьма точками на ньому. Після ініціації товару на нього завантажується AR у вигляді статичної чи анімаційної 3D моделі. Для коректного використання цієї технології, необхідно враховувати розміри усіх предметів на зображенні камери та їх положення. Таким чином розрахується необхідні дані про той об’єкт, який використовується за допомогою доповненій реальності.

При накладанні об’єктів на реальність є можливість відображати моделі (власно створені або з каталогу) на довільних предметах, які передаються через камеру; або на певному товару, зробивши його сканування для прив’язки певного об’єкту до товару.

Система, яка використовує технолгію доповненої реальності має наступні переваги, що є важливим для початку користування нею новими клієнтами:

- максимально зручний та дружній інтерфейс додатку;
- можливість створення своїх 3D-моделей;

– побудовані моделі, які готові для використання;

– можливість сканування безліч різноманітних об’єктів для відображення доповненої реальності;

– можливість легко і швидко накладати доповнену реальність на об’єкти.

Опишемо процес діяльності кожного з акторів системи. Ілюстратор створює 3D-моделі, завантажує їх у каталог додатку, та слідкує за якістю завантажених моделей користувачів. Користувач платформи сканує товари, накладає доповнену реальність на об’єкти, створює нові моделі за побажанням чи необхідністю. Користувач-компанія – сканує свої товари, накладає на них доповнену реальність та випускає у реліз для використання клієнтами. Адміністратор перевіряє ідентифікацію компаній, надає їм певного статусу та прав.

1.1.2. Опис функціональної моделі

Опис функціональної моделі представлено у вигдяді схеми структурної діаграми використання у частині дипломного проєкту «Графічний матеріал».

Акторами система є:

– ілюстратор;

– користувач;

– користувач-компанія;

– адміністратор.

Функціями ілюстратора є створення, редагування та видалення 3D-моделей. Також ілюстратор керує розподілом моделей по категоріям та їх якістю.

Користувач створює моделі, які йому подобаються, накладає своє власні чи з каталогу моделі на певний об’єкти чи на зображення з камери, сканує товари компаній для отримання додаткової інформації.

Користувач-компанія сканує товари своєї марки (можуть наноситись певні маркери для розуміння клієнтам, що цей товар можна сканувати), накладає на них 3D-моделі (також можна вибрати з каталогу, створити або завантажити власні), та публікує, даючи можливість тим самим при скані товару клієнтом – отримувати інформацію у вигляді AR.

Адміністратор взаємодіє з користувачем-компанією, перевіряє справжність існування такого підприємства, його цілі та задачі, надає особливі права. Також відповідає на питання та проводить підтримку сервісу.

1.2. Огляд наявних аналогів

В Україні на даний момент сфера доповненої реальності ще дуже не зайнята і тому компаній, які займаються цією технологією, дуже мало. У нашій країні 4 додатки AR.

На сьогоднішній день у світі нараховується 9 платформ, які працюють з доповненою реальністю у схожому форматі. Серед них додатки таких відомих компаній як Google та Microsoft. Можна виділити такі цікаві та корисні додатки як SketchAR, IKEA Place, Monster Park, Amazon. Головні сервіси, які займаються схожою тематикою з дипломним проєктом – Augment та Linzar, але вони пропонують лише частину можливостей доповненої реальності. Ці додатки також можуть сканувати товар, але у них немає можливості створення доповненої реальності, та не має можливості передавання інформації з доповненою реальністю звичайним користувачам. Також, завантаживши їх додатки та проаналізувавши, були зроблені висновки, що вони не відрізняються великою зручністю та легкістю у користуванні, відсутністю усіх необхідних функцій, якими має бути наповнений додаток для впровадження та сприйняття доповненої реальності. в порівнянні з додатком який буде реалізований у даному проєкті.

Не дивлячись на вже існуючу різноманітність систем з технологією доповненої реальності, розробники та компанії продовжують цим займатися,

адже при використанні цієї сфери з’являється більше можливостей покращити життя та зробити його легшим, підняти попит на послуги чи товари.

1.3. Постановки задачі

1.3.1. Призначення розробки

Призначення даного дипломного проєкту є полегшення впровадження об’єктів доповненої реальності у повсякденне життя роботи компаній та полегшення сприйняття їх клієнтами.

Цілі даної роботи:

- полегшення створення 3D-моделей;
- оптимізація етапу накладання 3D-моделей на об’єкти;
- полегшення управління доповненою інформацією для усіх товарів.

Система розробляється у вигляді мобільного додатку для забезпечення максимальної зручності користування таким сервісом.

1.3.2. Цілі та задачі розробки

Метою даного дипломного проєкту є автоматизація процесу впровадження доповненої реальності на підприємства та у життя людей, мінімізація вартості цієї дії та її спрощення за допомогою мобільного застосування.

В процесі написання диплому були поставлені наступні задачі:

- створення 3D-моделей;
- аналіз реального світу через камеру смартфона;
- накладання моделі на певний об’єкт з навколишнього середовища;
- вивід товару з доповненою реальністю на ринок.

Висновок по розділу

У даному розділі було розглянуте предметне середовище, процеси діяльності в системі, детально розглянуті ролі користувачів та методи їх роботи. Були описані цілі та мета, які ставляться перед розробкою системи, а також визначено задачі, вирішивши які система стане саме такою, як планується.

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Вхідні дані

Вхідними даними є інформація про учасника сервісу, 3D-моделі та предмети для сканування.

Інформація про учасника сервісу – це дані про нього та про його роль, а саме:

- електронна адреса користувача;
- пароль користувача для входу у систему;
- прізвище ім’я по-батькові;
- роль клієнта у додатку (звичайна людина або підприємство).

3D-моделі це об’єкти, які створені користувачами та додані у додаток, або відразу на предмет для сканування.

Предмети для сканування це предмети, на які компанія впроваджує моделі доповненої реальності та виводить на ринок.

2.2. Вихідні дані

До вихідних документів належать такі сторінки додатку:

- каталог моделей доповненої реальності, які будуть впровадженні у реальний світ;
- домашня сторінка користувача;
- предмети, які скануються через камеру;
- зображення, які утворюються накладанням доповненої реальності на об’єкти з реального світу через камеру.

Сторінка каталогу моделей доповненої реальності містить категорії моделей анімаційних зображень, а також редактор для побудови 3D-об’єкту.

Домашня сторінка користувача додатка містить наступну інформацію:

– електронну адресу користувача;
– прізвище ім’я по-батькові клієнта додатку;
– моделі, які створені даним користувачем;
– предмети, які були впровадженні з можливістю сканування.

2.3. Опис структури бази даних

Мобільний застосунок, що розробляється, побудовано на основі трьохрівненої клієнт-серверної архітектури, яка передбачає використання бази даних для зберігання інформації. Схема структурна бази даних представлена у частині диплому «Графічний матеріал».

Для більш детального опису таблиць бази даних нижче є описання призначення таблиць та їх даних.

Таблиця 1.1 – Перелік та призначення таблиць бази даних

Назва таблиці	Призначення
Users	Таблиця користувачів
Roles	Таблиця ролей
UserRoles	Таблиця ролей користувачів
Companies	Таблиця компаній, які використовують сервіс
UserCompanies	Таблиця користувачів з ідентифікацією компанії, в якій вони працюють
Categories	Таблиця категорій, на які розбиваються 3D-моделі
Models	Таблиця 3D-моделей
UserModels	Таблиця ідентифікації моделей за користувачами, які їх створили
Staff	Таблиця галузей предметів, які можна відсканувати і отримати інформацію у вигляді AR
ModelStaff	Таблиця прив’язаності моделей до предметів
UserStaff	Таблиця зв’язування предметів з користувачами, які їх додали

Більш детальна інформація вище приведених таблиць представлена у таблицях 1.2 – 1.12.

Таблиця 1.2 – Детальний опис таблиці ролей користувачів

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
UserRoles	UserId	Int	Ідентифікатор користувача
	RoleId	Int	Ідентифікатор ролі

Таблиця 1.3 – Детальний опис таблиці користувачів

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Users	Id	Int	Унікальний ідентифікатор
	FirstName	Nvarchar	Ім’я користувача
	LastName	Nvarchar	Прізвище користувача
	[Password]	Nvarchar	Пароль користувача
	Email	Nvarchar	Електронна почтова адреса користувача

Таблиця 1.4 – Детальний опис таблиці ролей

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Roles	Id	Int	Унікальний ідентифікатор
	Name	Nvarchar(max)	Назва ролі
	Description	Nvarchar(max)	Опис ролі

Таблиця 1.5 – Детальний опис таблиці компаній

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Companies	Id	Int	Унікальний ідентифікатор
	[Name]	Nvarchar(max)	Назва компанії
	ManagerId	Int	Ідентифікатор користувача, який є менеджером компанії у системі

Таблиця 1.6 – Детальний опис таблиці зв’язування користувачів та компаній

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
UserCompanies	CompanyId	Int	Ідентифікатор компанії
	UserId	Int	Ідентифікатор

			користувача

Таблиця 1.7 – Детальний опис таблиці категорій

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Categories	Id	Int	Унікальний ідентифікатор
	[Name]	Nvarchar(max)	Назва критерію
	CreatorId	Int	Ідентифікатор користувача, що створив критерій

Таблиця 1.8 – Детальний опис таблиці моделей

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Models	Id	Int	Унікальний ідентифікатор
	CategoryId	Nvarchar(max)	Ідентифікатор категорії
	ModelLink	Nvarchar(max)	Доступ до файлу з моделлю

Таблиця 1.9 – Детальний опис таблиці користувачів зі створеними моделями

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
UserModels	UserId	Int	Ідентифікатор користувача
	ModelId	Int	Ідентифікатор моделі

Таблиця 1.10 – Детальний опис таблиці предметів для сканування

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Staff	Id	Int	Ідентифікатор користувача
	StaffLink	Nvarchar(max)	Доступ до файлу з предметами для сканування

Таблиця 1.11 – Детальний опис таблиці моделей з предметами

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
ModelStaff	ModelId	Int	Ідентифікатор моделі
	StaffId	Int	Ідентифікатор предмету

Таблиця 1.12 – Детальний опис таблиці предметів користувачів

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
UserStaff	UserId	Int	Ідентифікатор користувача
	StaffId	Int	Ідентифікатор предмету

Висновок до розділу

В даному розділі було розглянуто вхідні та вихідні дані для системи. Також, було описано структуру бази даних. Вона налічує 12 таблиць, детальний опис яких представлено у вигляді таблиць.

Побудовано схему структурну бази даних та приведено опис її структури у табличному форматі.

3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Змістовна постановка задачі

Маємо наступні функції, які повинна виконувати система:

- Надавати можливість переглядати готові 3D-моделі;
- Накладати об’єкти на зображення з камери смартфона;
- Створювати нові моделі;
- Публікувати у мережу предмети з доповненою реальністю;
- Сканувати предмети з камери і відтворювати AR.

Отже, задача полягає у реалізації алгоритму накладання моделей доповненої реальності на зображення з камери, та алгоритму ідентифікації предметів з реального світу. Для побудови доповненої реальності на відео-потоці відбувається пошук маркерів та відображення 3D-моделі на ньому. Щоб користувачі могли відсканувати предмети з реального світу, компанії, які будуть клієнтами сервісу, повинні будуть зробити фото цього предмету. Далі у файл у базу даних записуються всі точки, по яким пізніше буде проходити ідентифікація цього товару. Коли користувач запускає камеру, то вхідні дані з нею порівнюються з існуючими даними для пошуку схожих предметів. Після знаходження програмою даного товару, запускається потік відображення об’єктів доповненої реальності. Тут постає друга задача: накласти цифрову інформацію таким чином, щоб вона виглядала гармонійно на фоні реального світу. Для цього нам необхідно брати розміри та положення об’єкту, який накладається, та предметів з камери. Потім відбувається зіставлення цих двох даних та відображення.

3.2. Математична постановка задачі

Нехай існує матриця А моделі доповненої реальності, яка складається з її розміру, розтягування та положення, та матриця В реального світу з камери смартфона, яка складається з фокусної відстані, кута обзору та розміри.

Ціллю є визначити, де і яким чином необхідно відобразити 3D-модель доповненої реальності, щоб вона зливалася зі світом та виглядала гармонійно.

Нехай існує зображення з ділянками L, на яких є особливі точки D(x). Ціллю є визначити, який предмет з зображення є необхідним для накладання доповненої реальності.

3.3. Обґрунтування методу розв'язання

Метод розв'язання проєкту полягає у декількох діях. По-перше, необхідно провести розпізнавання реальних об'єктів. Для цього програма буде виділяти точки та окружності, використовуючи алгоритм SIFT на основі підходу Feature Based. Алгоритм шукає особливі частини об'єктів та записує їх. По-друге, програма повинна накласти модель на реальний світ. Це робиться завдяки OpenGL та побудови матриць взаємодії моделі та зображення з камери смартфона.

3.4. Опис методів розв'язання

Реалізація алгоритму полягає у наступному:

– Знаходження особливих точок.

Побудова піраміди Гаусіанів (зображення, що розмите фільтром Гауса) – увесь простір розбивається на деякі ділянки – октави (кожна наступна у два рази більша за попередню):

[11]

, де – значення Гаусіана в точці з координатами , – радіус розмиття, – Гаусове ядро, – значення вихідного зображення, * – операція згортки.

Гаусове ядро зі степенем розмиття знаходиться за формулою:

—— [11]

Побудова різниць Гаусіанів (зображення, яке отримане шляхом піксельного віднімання Гаусіана вихідного зображення від Гаусіана з іншим радіусом розмиття):

[11]

Далі шукаємо особливі точки – точки, які є локальними екстремумами різниці Гаусіанів.

– Фільтрація ключових точок.

Обчислюємо особливі точки з субпиксельною точністю за допомогою апроксимування функції різниць Гаусіанів многочленом Тейлора другого порядку у точці екстремуму:

— – — [11]

, де – вектор зміщення відносно точки розкладання, перша похідна – градієнт, друга похідна – матриця Гессе.

Екстремум многочлена Тейлора знаходиться за формулою:

———

Отже, отримуємо СЛАУ 3x3 відносно компонент вектору . Якщо одна з компонент більша за половину кроку, то це означає, що точка обрана неправильно і необхідно йти до сусідньої точки. Потім все повторюється. Якщо виходимо за границі – видаляємо цю точку.

Видалення точок з малим контрастом за формулою:

– — [11]

Видалення точок на границі об’єктів. Для перевірки застосуємо матрицю Гессе 2x2:

Нехай – слід матриці, а – її визначник. Тоді:

[11]

Нехай – відношення великого згину до меншого:

Тоді:

[11]

Точка не видаляється, якщо:

[11]

– Обчислення орієнтації ключової точки. Напрямок ключової точки знаходиться з напрямків градієнтів сусідніх точок. Обчислити величину і напрямок градієнту у точці можна відповідно за формулами:

[11]

Далі будується гістограма градієнтів, обирається напрямок, який відповідає максимальній компоненті гістограми, присвоюються точці усі напрямки, яким відповідають значення

– Дескриптор отримується з усіх отриманих гістограм, потім нормалізується та тепер вони готові для використання.

Висновок до розділу

У даному розділі було сформовано змістовну постановку задачі, визначено функції, які система повинна виконувати, а саме: можливість переглядати готові 3D-моделі, накладати об’єкти на зображення з камери смартфона, створювати нові моделі, публікувати у мережу предмети з доповненою реальністю, сканувати предмети з камери і відтворювати AR.

Описано алгоритм для ідентифікації зображень та обґрунтовано даний метод розв’язання.

4. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Засоби розробки

Реалізація додатку була виконана за допомогою мови програмування Java з використанням офіційного середовища розробки Android Studio.

Java - об'єктно-орієнтована мова програмування. Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині незалежно від комп'ютерної архітектури [6].

Java - дуже гнучка мова з відкритим вихідним кодом. Для розробки програми існують архітектурні підходи до проєкту, які вирішують проблему розробки стандартного програмного забезпечення [6].

Вперше ця мова програмування була випущена у 1995 році компанією Sun Microsystems. Наразі вона завжди на перших місцях по використанню її у світі. Все це завдяки її швидкості, рівнем захисту та надійністю. Java використовується для комп’ютерів, центрів даних, гральних консолей, суперкомп’ютерів, телефонів, інтернету та багато іншого.

Також для описання розмітки зовнішніх сторінок у додатку використовується XML (Extensible Markup Language) – розширювана мова розмітки. Ця мова задає правила та стандарти для розмітки даних на сторінках з використанням тегів. Завдяки цим тегам XML каже програмі де і яким чином розмістити певну інформацію.

Android Studio – офіційна IDE (інтегроване середовище розробки), яке створене компанією Google для розробки мобільних додатків на базі Android [7].

У якості бази даних для мобільного додатку використовується база даних Firebase. Дана платформа має багато служб, тож є дуже зручною для

додатку, що розробляється, адже можна зберігати файли на дисковому просторі у хмарі на базі Firebase, та застосовувати базу даних. Завдяки використанню однієї і тієї ж платформи для цих двох дій можна забезпечити більш швидку передачу даних. Також база даних Firebase дозволяє використовувати дані у режимі реального часу для користувачів, що є практично при використанні даних, які можуть одночасно обновлятися одними користувачами та використовуватися іншими. [8]

Усі описані засоби розробки є безкоштовними та найбільш адаптованими для розробки мобільних застосунків. Тому та з описаних вище переваг було обрано ці технології та програми.

4.2. Вимоги до технічного забезпечення

Вимоги до технічного забезпечення встановленні шляхом визначення можливостей для забезпечення виконання операцій процесів програми та коректної їх обробки.

4.2.1. Загальні вимоги

Для коректної роботи мобільного додатку, необхідно мати смартфон з наступною конфігурацією технічних засобів:

При використанні мобільного додатку, пропонується смартфон, який може мати приблизно наступні характеристики:

- процесор з тактовою частотою не нижче 1,6 ГГц;
- оперативна пам’ять (не менше 1024 МБ);
- об’єм пам’яті смартфона не нижчий за 2Гб;
- доступ до камери;
- доступ до мережі інтернет.

Виходячи з того, що Firebase – це хмарна база даних, при розробці програмного забезпечення не потрібно настроювати сервер під базу даних та

хвилюватися за його конфігурацію. Для мобільного додатку, що розробляється, необхідно, щоб база даних мала щонайменше:

– 1 GB дискового простору;
– 100 одночасних підключень;
– скачування 10 GB у місяць;
– 20 тисяч операцій загрузки даних;
– 50 тисяч операцій скачування даних.

Після впровадження мобільного додатку для його загрузки необхідно мати на смартфоні PlayMarket та обліковий запис Google для можливості скачування додатків.

4.3. Архітектура програмного забезпечення

Мобільний додаток, що розробляться, має трирівневу архітектуру. Трирівнева архітектура забезпечує взаємодію трьох компонентів: сервера для зберігання даних, логіку виконання процесів та зовнішній вигляд системи.

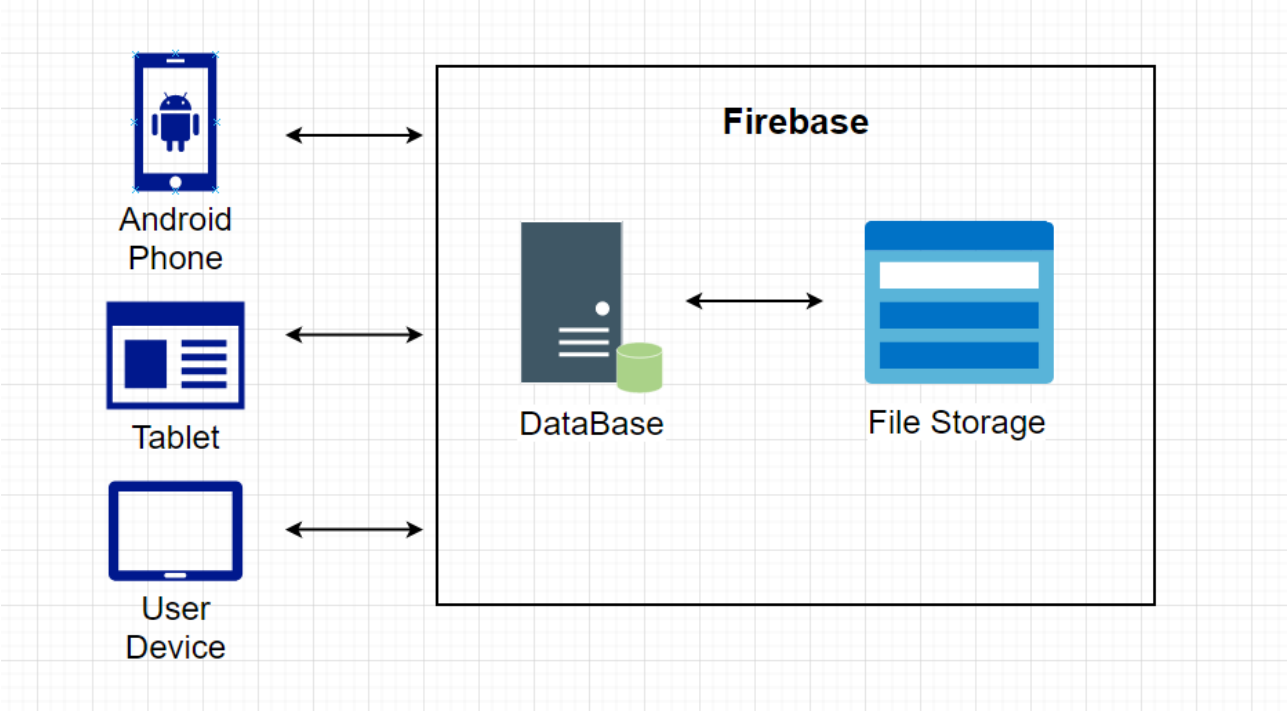


Рисунок 4.1 – Архітектура програмного забезпечення

4.3.1. Діаграма класів

У частині дипломного проєкту «Графічний матеріал» представлена діаграма класів системи, що стосується модуля ідентифікації зображення.

Призначення зображених вище класів наведено в таблиці 4.1.

Таблиця 4.1 – Призначення класів системи

Назва класу	Призначення
CameraActivity	Опис роботи камери
ClassifierActivity	Опис роботи активності класифікатора
BorderedText	Перетворення предметів на текст
CameraConnectionFragment	Отримує зображення з камери
Logger	Виконує логування
Classifier	Класифікатор – маркує зображення
ClassifierFloatMobileNet	Реалізація класифікатора з плаваючою моделлю
ClassifierFloatEfficientNet	Реалізація класифікатора з плаваючою моделлю

4.3.2. Специфікація функцій

В наступних таблицях описані методи класів системи. Наведені класи відносяться до модуля ідентифікації зображень.

Таблиця 4.2 – Опис методів класу CameraActivity

Метод	**Опис методу**	**Параметри**	**Опис параметрів**
onCreate	Базовий метод для запуску мобільного додатку	savedInstance State	Базовий параметр для запуску додатку
getRgbBytes	Отримання картинки у байтовому коді	не передбачені	не передбачені
getLuminance Stride	Отримання світлової яскравості	не передбачені	не передбачені

getLuminance	Отримання яскравості	не передбачені	не передбачені
onPreviewFrame	Виклик API камери	bytes	Вміст кадру у форматі
		camera	Об’єкт стандартного класу Camera
onImageAvailable	Виклик API другоїкамери	reader	Читання картинки з яким пов’язаний виклик
onStart	Викликається при явному запуску послуги	не передбачені	не передбачені
onResume	Відновлення взаємодії з користувачем	не передбачені	не передбачені
onPause	Викликається при зупинці використання послуги, але вона ще присутня	не передбачені	не передбачені
onStop	Викликається при втраті користувача	не передбачені	не передбачені
onDestroy	Очищає не потрібні служби	не передбачені	не передбачені
runInBackground	Робота у фоновому режимі	runnable	
onRequestPermissionsResult	Отримання результатів для запитів на дозвіл	requestCode	Код запиту передано
		permissions	Запитувані дозволи
		grantResults	Результати надання грантів

isHardwareLevelSupported	Перевірка рівня технічного забезпечення смартфона	characteristics	Характеристики, отримані зі смартфона
		requiredLevel	Значення необхідного рівня
chooseCamera	Вибір передньої чи задньої камери	не передбачені	не передбачені
allPermissionsGranted	Усі права назначені	grantResults	Результати грантів
hasPermission	Наявність прав	не передбачені	не передбачені
requestPermission	Запит прав	не передбачені	не передбачені
chooseCamera	Вибір камери	не передбачені	не передбачені
setFragment	Задання фрагменту сторінки для камери	не передбачені	не передбачені
fillBytes	Метод для заповнення байтів	planes	Масив вхідних байтів
		yuvBytes	Масив ємностей масивів байтів
readyForNextImage	Перевірка на готовність для наступної картинки	не передбачені	не передбачені
getScreenOrientation	отримує орієнтацію смартфона	не передбачені	не передбачені
showResultsInBottomSheet	Відображення результату	results	Вхідний результат

Таблиця 4.3 – Опис методів класу ClassifierActivity

Метод	Опис методу	Параметри	Опис параметрів
getLayoutId	Отримує розмітку	не передбачені	не передбачені
getDesiredPreviewFrameSize	Отримує розмір вікна	не передбачені	не передбачені
onPreviewSizeChosen	Вибір оптимального розміру передперегляду	size	Розмір
		rotation	Кут обертання
processImage	Опрацьовує зображення	не передбачені	не передбачені
onInferenceConfigurationChanged	Призначення даних для створення умовивіду	не передбачені	не передбачені
recreateClassifier	Відтворення класифікатору	model	Тип моделі
		device	Тип пристрою виконання
		numThreads	Кількість потоків

Таблиця 4.4 – Опис методів класу BorderedText

Метод	Опис методу	Параметри	Опис параметрів
BorderedText	Конструктор для створення тексту з певним кольором, розміром та розміщенням	textSize	Розмір тексту
BorderedText	Конструктор для створення тексту з певним кольором,	interiorColor	Колір тексту
		exteriorColor	Колір заливки
		textSize	Розмір тексту

	розміром та розміщенням		
drawText	Відображення тексту	canvas	Об’єкт класу Canvasдля малювання
		posX	Позиція по осі X
		posY	Позиція по осі Y
		text	Текст для відображення
drawText	Відображення ліній	canvas	Об’єкт класу Canvasдля малювання
		posX	Позиція по осі X
		posY	Позиція по осі Y
		lines	Лінії для відображення
setInteriorColor	Задання кольору тексту	color	Колір
setExteriorColor	Задання кольору заливки	color	Колір
getTextSize	Отримання розміру тексту	не передбачені	не передбачені

Таблиця 4.5 – Опис методів класу CameraConnectionFragment

Метод	**Опис методу**	**Параметри**	**Опис параметрів**
CameraConnectionFragment	Конструктор для визначення параметрів	connectionCallback	Об’єкт класу ConnectionCallback
		imageListener	Реалізація OnImageAvailableListener

		layout	Значення ідентифікаційного номеру сторінки
		inputSize	Вхідний розмір
chooseOptimalSize	Вибір найоптимальнішого розміру	choices	Розміри, що підтримуються камерою
		width	Ширина
		height	Висота
showToast	Відображення тексту	text	Текст для відображення
setUpCameraOutputs	Налаштування змінних членів, що пов’язані з камерою	не передбачені	не передбачені
openCamera	Відкриття камери	width	
		height	
closeCamera	Закриття камери	не передбачені	не передбачені
startBackgroundThread	Початок потоку у фоновому режимі	не передбачені	не передбачені
createCameraPreviewSession	Створення нової сесії для передперегляду камери	не передбачені	не передбачені
configureTransform	Налаштовує необхідне перетворення	viewWidth	Ширина відображення текстури
		viewHeight	Висота відображення

			текстури

4.3.3. Діаграма послідовності

Для опису дій накладання об’єктів на реальний світ було побудовано діаграму послідовності, яку предсталено у частині дипломного проєкту «Графічний матеріал»

Висновок до розділу

У даному розділі було визначено за допомогою яких засобів реалізації створюється додаток, а також яку конфігурацію повинен мати засіб для користування додатком – смартфон чи планшет. Була побудована діаграма класів, розписана специфікація функцій, які використовуються для ідентифікації зображень.

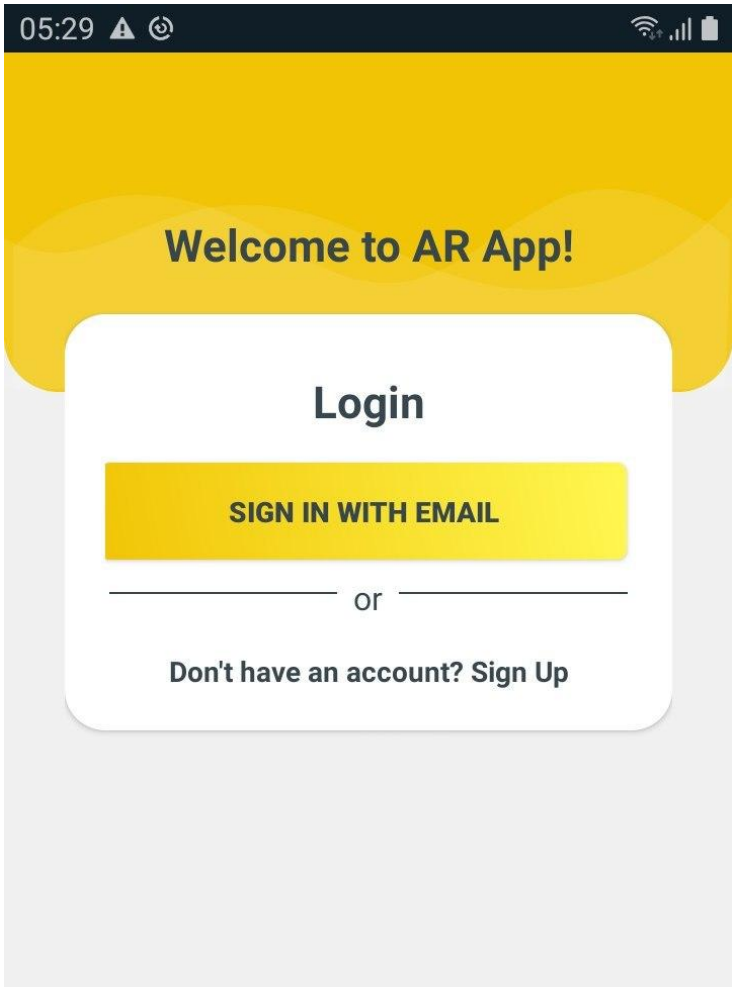
5. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1. Керівництво користувача

У даному розділі наведене керівництво користувача інформаційної системи удосконалення сприйняття інформації за топомогою технології доповненої реальності.

На початку роботи, користувач повинен авторизуватись. Для йому потрібно ввести свою адресу електронної пошти та пароль, який він використовував при реєстрації.

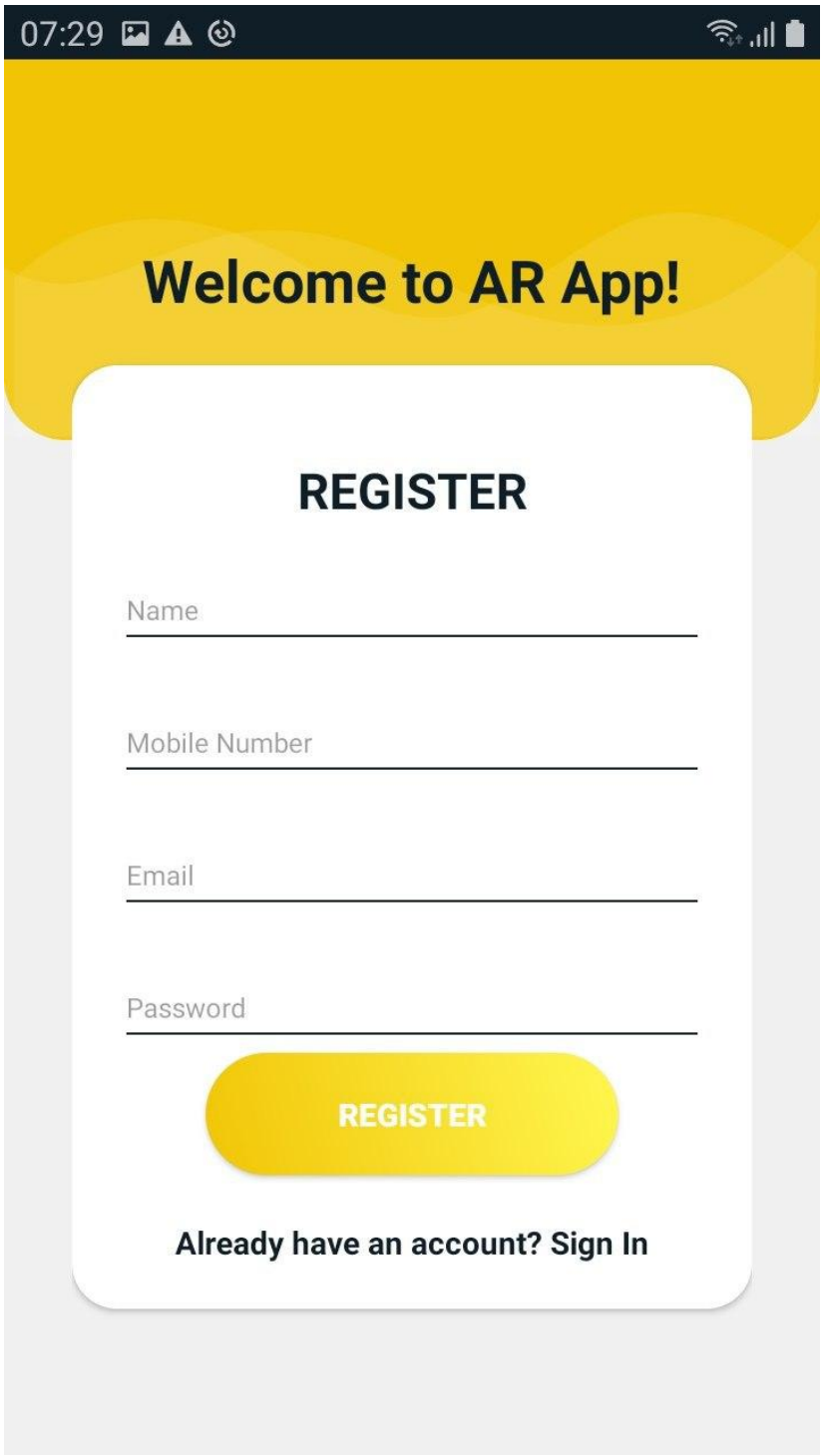
Сторінка авторизація виглядає таким чином, як зображено на рисунку



5.1.

Рисунок 5.1 – Сторінка авторизації

Якщо користувач ще не зареєстрований, є можливість перейти на сторінку реєстрації та ввести ПІБ, адресу електронної пошти, мобільний



номер та пароль. Сторінка реєстрації зображена на рисунку 5.2.

Рисунок 5.2 – Сторінка реєстрації користувачів

Після того, як користувач увійшов у систему, у нього з’являється можливість бачити та використовувати моделі з каталогу для побудови доповненої реальності.

На рисунку 5.3 зображено скріншот сторінки каталогу 3D-моделей, які вже існують у базі даних додатку та які можна використовувати.

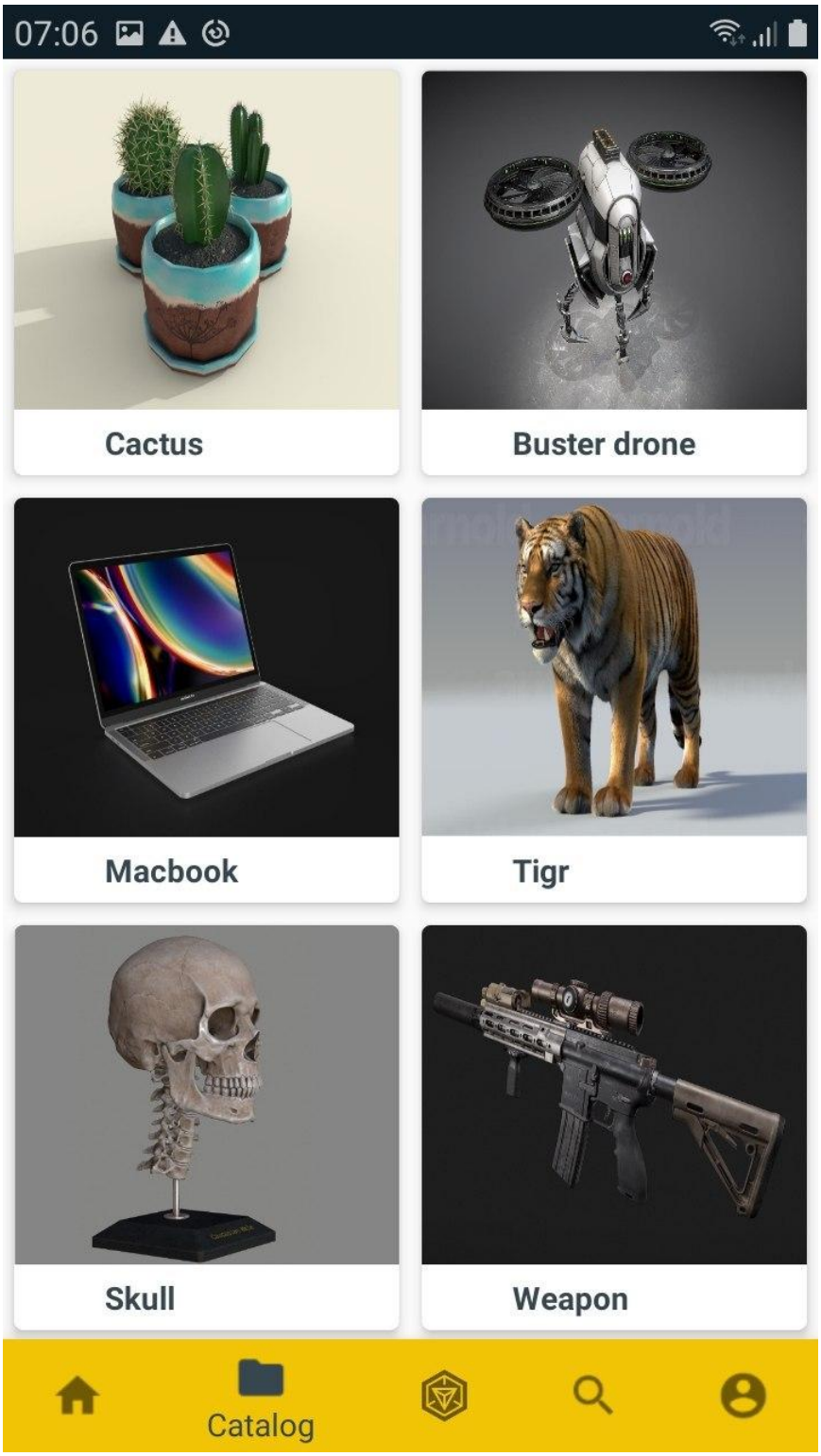


Рисунок 5.3 – Каталог 3D-моделей

Користувач може обрати будь-яку модель, яка його цікавить та спробувати її використати – накласти на реальний світ. Для цього спочатку користувач нажимає на модель, що йому сподобалася, та далі він перейде на сторінку з інформацією про цю модель та збільшеним її зображенням. На рисунку 5.4 зображено сторінку з інформацією про модель.

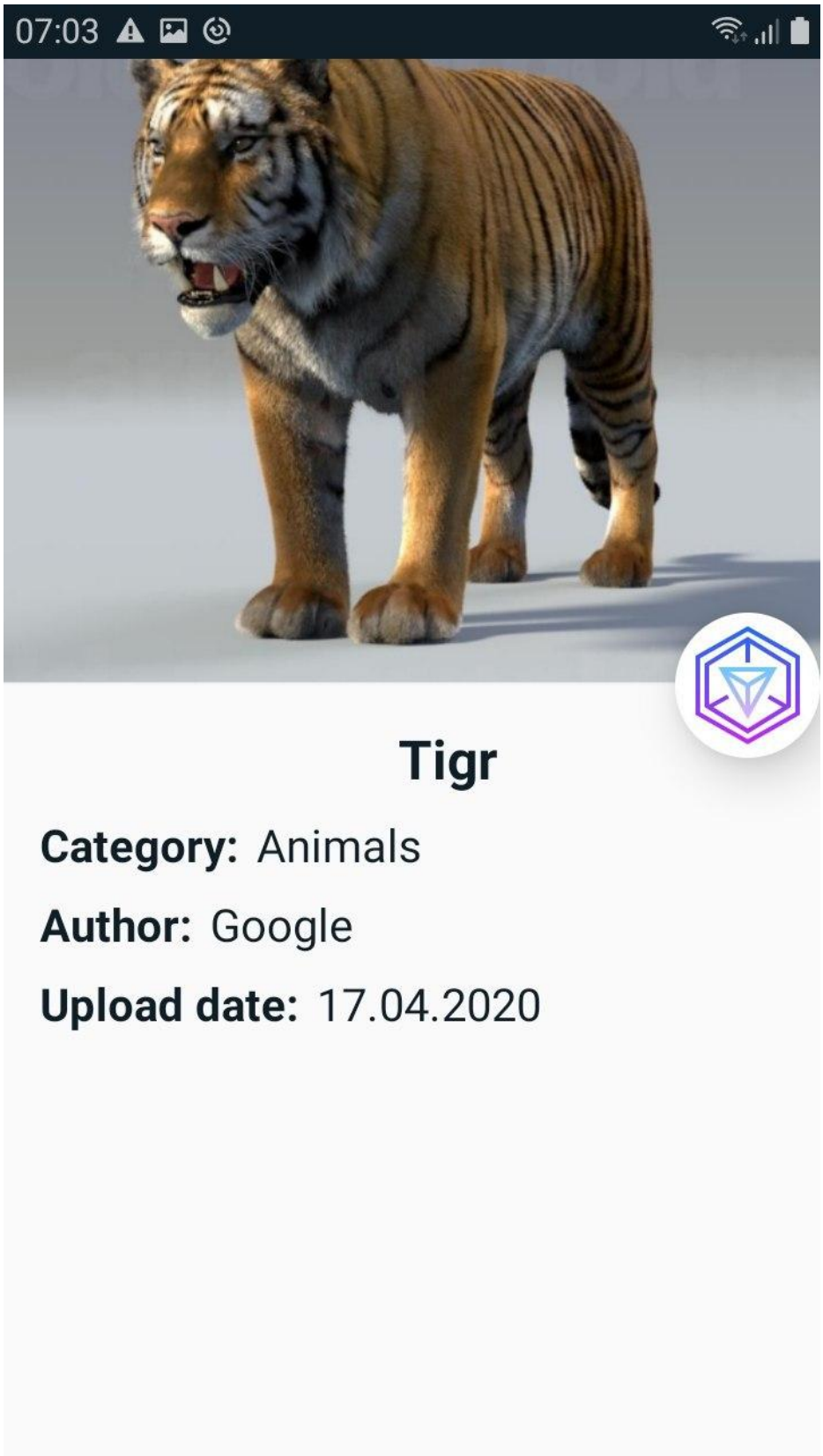
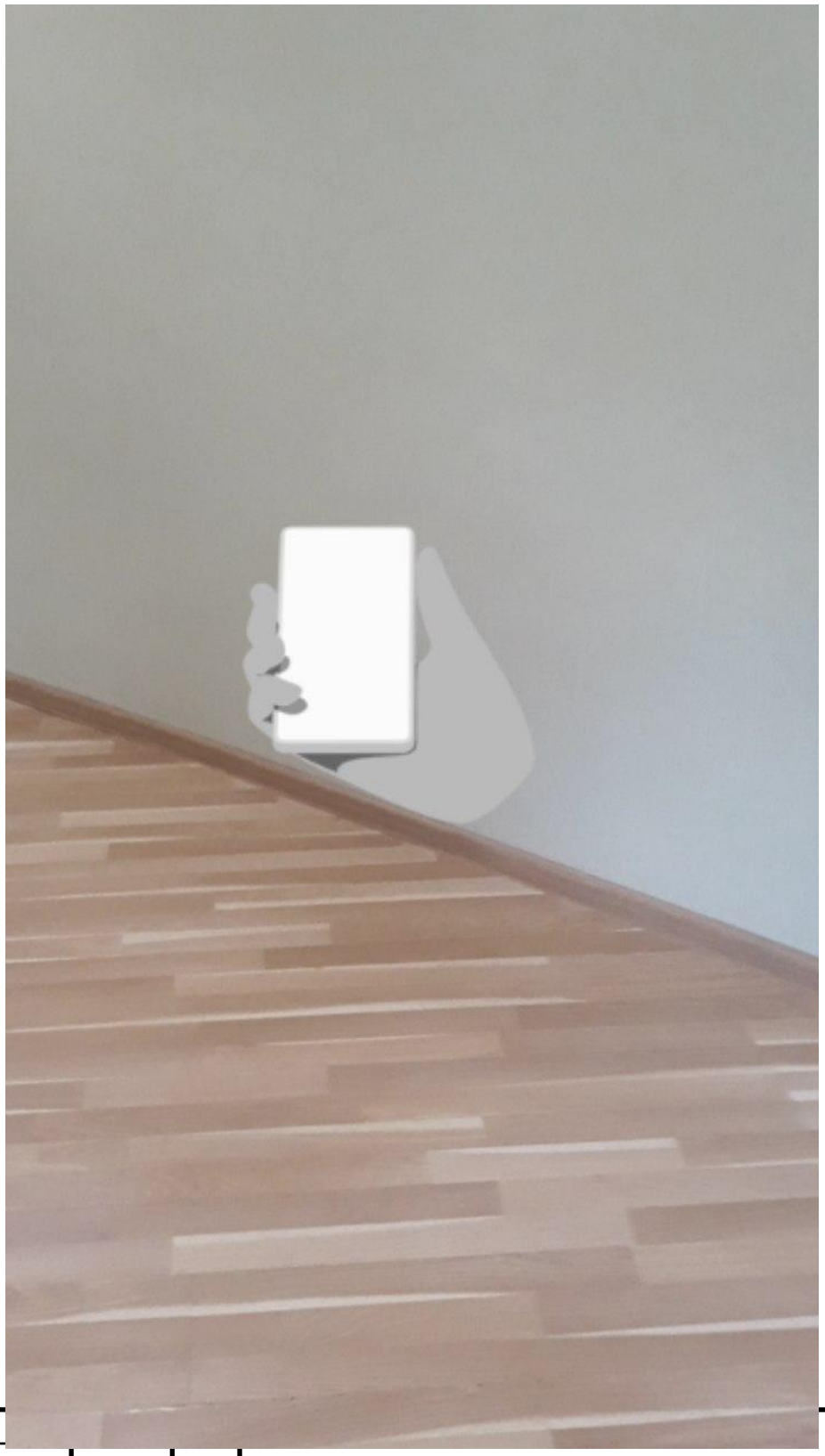


Рисунок 5.4 – Сторінка з інформацією про модель

Наступним кроком є включення камери та накладання моделі. Для цього необхідно, щоб користувач натиснув кнопку, що знаходиться праворуч

між зображенням моделі та описанням. На рисунку 5.4 можна побачити дану кнопку.

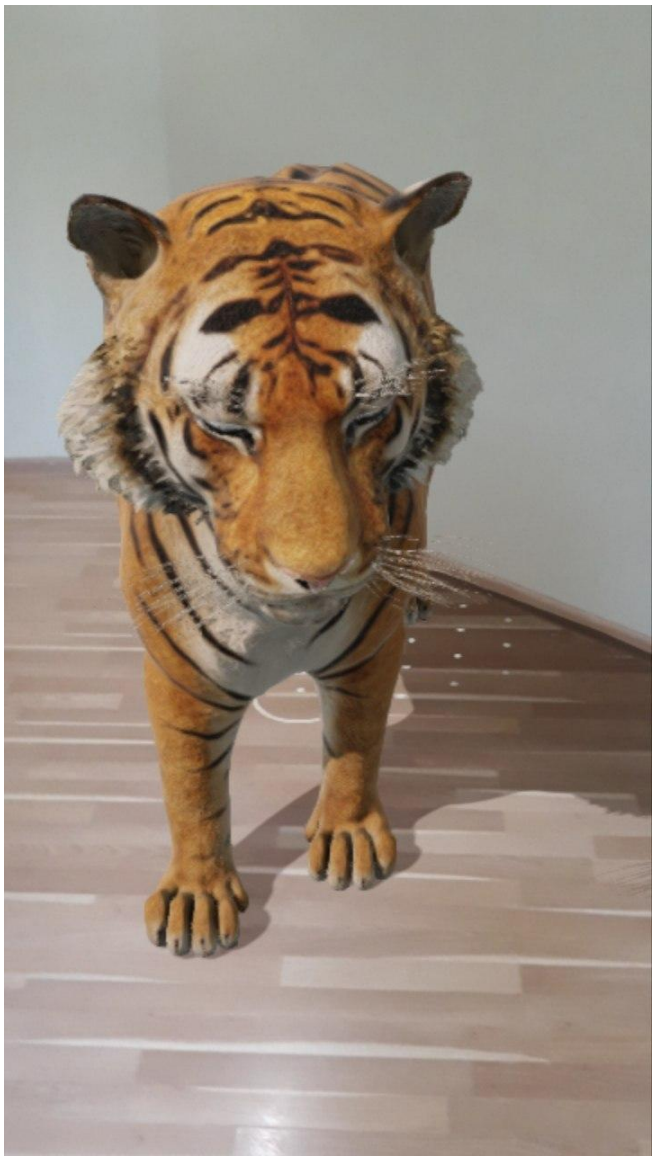
Після натискання користувач перейде на сторінку, на яку транслюється потік зображення з камери смартфона. Це буде виглядати таким чином, як на



рисунку 5.5.

Рисунок 5.5 – Потік зображення з камери смартфона

Як видно зі скріншоту, програма показує легкі оберти рукою задля побудови точок площини, на яку можна накласти 3D-модель. Після отримання інформації про навколишній світ та положення смартфона відносно простору, додаток відображає модель. Це показано на рисунках 5.6-5.7.



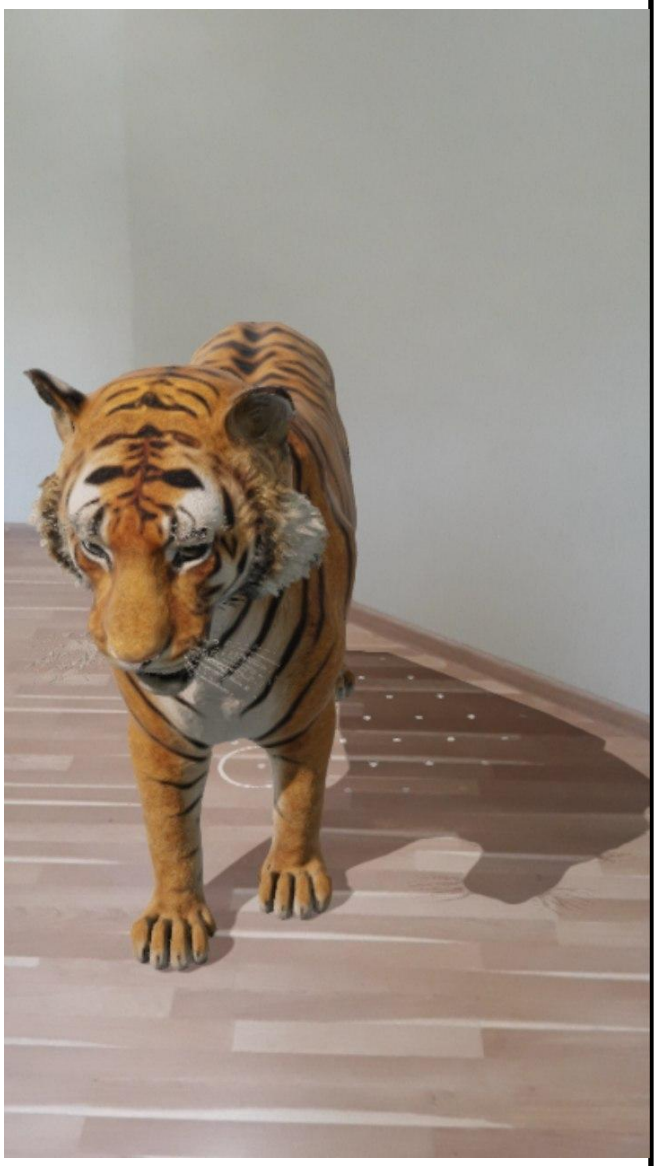


Рисунок 5.6-5.7 – Відображення моделі у світі

Технологія доповненої реальності у маємо додатку може бути використана у різних цілях підприємств. Наприклад, компанія по виготовленню та продажі мебелів може публікувати свої товари у вигляді

3D-моделей, а користувачі можуть проектувати їх на реальність у оселях та обирати ті, що найбільше їм підходять (рис 5.8.).

Рисунок 5.8 – Відображення мебелі у квартирі

5.2. Випробування програмного продукту

Програма і методика випробувань містять опис тестів і порядок їх виконання для перевірки відповідності функціональним вимогам, представленим у технічному завданні на створення інформаційної системи з удосконалення сприйняття інформації технології доповненої реальності.

При проведенні випробувань перевіряється відповідність додатку до вимог, що були поставлені на початку, у технічному завданні на створення інформаційної системи з удосконалення сприйняття інформації технології доповненої реальності.

Для цього було проведені різноманітні тести, які нижче детально описані у вигдяді таблиць 5.1 -5.11.

Випробування, які проводяться над системою, що розробляється, є ручними по автоматизації та інтеграційним по ізольованості компонентів. Якщо брати до уваги знання системи, то зроблені випробування належать до випробувань чорного ящику.

5.2.1. Мета випробувань

Метою випробувань є перевірка на функціональність інформаційної системи з удосконалення сприйняття інформації технології доповненої реальності.

5.2.2. Загальні положення

В процесі випробуання застосовувались наступні документи:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем [12].
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів[13].

5.2.3. Результати випробувань

Під час тестування була зроблена перевірка роботи програми на функціональність інформаційної системи удосконалення сприйняття інформації технології доповненої реальності.

Тести, які використовувались, в процесі випробування представлені у таблицях 5.1 – 5.11.

Таблиця 5.1 – Тест вдалого проходження авторизації користувача

Мета тесту	Перевірити спроможність авторизації користувачів
Початковий стан	Клієнт додатку, який вже існує у системі як зареєстрований користувач, бачить сторінку входу
Вхідні данні	Адреса електронної пошти та пароль користувача
Схема проведення тесту	Користувач вносить необхідну інформацію у поля для неї.
Очікуваний результат	Користувача пройшов авторизацію та його перенаправлено до його початкової сторінки авторизованих користувачів.
Стан після проведення випробувань	Користувач увійшов до системи та бачить каталог моделей.

Таблиця 5.2 – Тест невдалого проходження авторизації користувача

Мета тесту	Перевірити спроможність авторизації користувачів
Початковий стан	Клієнт додатку, який вже існує у системі як зареєстрований користувач, бачить сторінку входу
Вхідні данні	Не правильні адреса електронної пошти або пароль
Схема проведення тесту	Користувач вносить необхідну інформацію у поля для неї.
Очікуваний результат	Користувач не пройшов авторизацію та не допущений у систему
Стан після проведення випробувань	Користувач не увійшов до системи. Виведено повідомлення про некоректну введену інформацію та пропозицію ввести ще раз.

Таблиця 5.3 – Тест реєстрації в системі

Мета тесту	Перевірка можливості зареєструватися у системі
Початковий стан	Клієнт додатку бачить сторінку реєстрації

Вхідні данні	Адреса електронної пошти, ПІБ, роль та пароль
Схема проведення тесту	Користувач вводить необхідну інформацію у поля для цього та натискає кнопку «Зареєструватись»
Очікуваний результат	Користувача зареєстровано в системі та перенаправлено на сторінку для входу у систему.
Стан після проведення випробувань	Виводиться повідомлення про успішну реєстрацію та відкривається сторінка для входу в систему

Таблиця 5.4 – Тест невдалої реєстрації в системі

Мета тесту	Перевірка можливості зареєструватися у системі
Початковий стан	Клієнт додатку бачить сторінку реєстрації
Вхідні данні	Адреса електронної пошти, яка вже зареєстрована у системі, ПІБ, роль та пароль
Схема проведення тесту	Користувач вводить необхідну інформацію у поля для цього та натискає кнопку «Зареєструватись»
Очікуваний результат	Невдала реєстрація в системі
Стан після проведення випробувань	Відкривається сторінка входу у систему та показано повідомлення про те, що користувач з даною електронною поштою вже зареєстрований

Таблиця 5.5 – Тест невдалої реєстрації в системі

Мета тесту	Перевірка можливості зареєструватися у системі
Початковий стан	Клієнт додатку бачить сторінку реєстрації
Вхідні данні	Адреса електронної пошти, ПІБ, роль, пароль та його повторення, яке не співпадає з першим разом
Схема проведення тесту	Користувач вводить необхідну інформацію у поля для цього та натискає кнопку «Зареєструватись»
Очікуваний результат	Невдала реєстрація в системі
Стан після проведення	Користувача не зареєстровано у системі, та показано

випробувань	повідомлення про те, що користувач ввів різні паролі та пропозиція виправити це

Таблиця 5.6 – Тест невдалої реєстрації в системі

Мета тесту	Перевірка можливості зареєструватися у системі
Початковий стан	Клієнт додатку бачить сторінку реєстрації
Вхідні данні	Адреса електронної пошти, яка неможлива, ПІБ, роль та пароль
Схема проведення тесту	Користувач вводить необхідну інформацію у поля для цього та натискає кнопку «Зареєструватись»
Очікуваний результат	Невдала реєстрація в системі
Стан після проведення випробувань	Користувача не зареєстровано та показано повідомлення про те, що електронна пошта вказана не правильно

Таблиця 5.7 – Тестування можливості виходу з додатку

Мета тесту	Перевірка виходу користувача із додатку
Початковий стан	Авторизований користувач знаходиться на будь якій сторінці додатку
Вхідні данні	
Схема проведення тесту	Користувач натискає на кнопку «Вийти»
Очікуваний результат	Користувач вийшов із системи
Стан після проведення випробувань	Користувач вийшов із системи і бачить сторінку для входу

Таблиця 5.8 – Тест модуля перегляду інформації про 3D-модель

Мета тесту	Перевірка можливості користувача переглядати інформацію про моделі
Початковий стан	Авторизований користувач знаходиться на сторінці

	всіх моделей
Вхідні данні	
Схема проведення тесту	Користувач натискає на бажану модель
Очікуваний результат	Відображається докладна інформація про модель та її збільшене зображення
Стан після проведення випробувань	Переправляється користувач на сторінку, де представлена докладна інформація про модель та її збільшене зображення.

Таблиця 5.9 – Тест накладання моделі на реальний світ

Мета тесту	Перевірити накладання 3Dмоделі на зображення з камери смартфона
Початковий стан	Користувач знаходиться на сторінці інформації про модель
Вхідні данні	
Схема проведення тесту	Натиснути на кнопку «Накласти»
Очікуваний результат	Модель відображена на площині
Стан після проведення випробувань	Модель знаходиться на площині, на яку була наведена камера

Таблиця 5.10 – Тест пересування моделі у просторі

Мета тесту	Перевірити модуль пересування моделі у реальному часі на зображенні з камери
Початковий стан	Користувач знаходиться на зображенні з камери з відображеною доповненою реальністю
Вхідні данні	3Dмодель відображена на зображенні з камери
Схема проведення	Пересувати модель у просторі

тесту	
Очікуваний результат	Модель пересувається за пальцем користувача
Стан після проведення випробувань	Модель була пересунута на інше місце

Таблиця 5.11 – Тест зміни розміру моделі на зображенні з камери

Мета тесту	Перевірити модуль зміни розміру моделі на зображенні з камери смартфона
Початковий стан	Користувач знаходиться на зображенні з камери з відображеною доповненою реальністю
Вхідні данні	3Dмодель відображена на зображенні з камери
Схема проведення тесту	Двома пальцями зменшувати та збільшувати розміри моделі
Очікуваний результат	Розмір моделі змінено
Стан після проведення випробувань	Модель стала більша/менша

Висновок до розділу

У даному розділі описано кроки роботи з програмою та представлено керівництво користувача зі скріншотами роботи цієї програми. Також описане тестування, яке проводиться над системою для перевірки її коректної роботи та відповідність вимогам.

ЗАГАЛЬНІ ВИСНОВКИ

Під час роботи над дипломним проєктом було визначено цілі та задачі роботи, а також методи її реалізації. Було розглянуто алгоритм ідентифікації зображень та накладання доповненої реальності на зображення з камери смартфона.

Метою створення інформаційної системи є удосконалення сприйняття інформації технології доповненої реальності, а також полегшення впровадження та використання доповненої реальності у житті та на підприємствах.

В описі інформаційного забезпечення описана мова програмування Java, середовище розробки проєкту AndroidStudio, представлена база даних Firebase. Була побудована діаграма бази даних для додатку.

Було визначено алгоритм для реалізації ідентифікації зображень та сформульовано математичну модель по його крокам.

При описі архітектури програмного забезпечення, було показано, що додаток має клієнт-серверну архітектуру, адже для отримання об’єктів та реалізації авторизації необхідна база даних. Також визначено вид тестування та описано його результати. Система відповідає заявленим вимогам. Наведено інструкцію користувача по роботі із модулем накладання доповненої реальності на камеру смартфона за допомогою мобільного додатку.

Розроблена інформаційна системи з удосконалення сприйняття інформації технології доповненої реальності за допомогою мобільного додатку допомагає компаніям впроваджувати доповнену реальність на свої товари для покращення продажу та збільшення попиту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Швець Д.Ю. Алгоритми створення доповненої реальності. Матеріали IV всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 24-30 квітня 2020 р. – С. 160.
2. Zitova B. Image registration methods: a survey / B. Zitova, J. Flusser. // Image and Vision Computing. – 2003. – №21. – С. 97–1000.
3. Трехуровневая клиент-серверная архитектура [Електронний ресурс] // Режим доступу: https://studopedia.ru/7_95327_trehurovnevaya-klient-servernaya-arhitektura.html
4. Google Developers Launchpad [Електронний ресурс] // Режим доступу: https://developers.google.com/programs/launchpad/
5. What is Augmented Reality? [Електронний ресурс] // Режим доступу: https://www.interaction-design.org/literature/topics/augmented-reality
6. Java [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Java
7. Android Studio [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
8. Firebase [Електронний ресурс] // Режим доступу: https://firebase.google.com/
9. Collecting Data for Custom Object Detection [Електронний ресурс] // Режим доступу: https://towardsdatascience.com/collecting-data-for-custom-object-detection-e7d888c1469b

10. Object detection with deep learning and OpenCV [Електронний ресурс] // Режим доступу: https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/
11. Построение SIFT дескрипторов и задача сопоставления изображений [Електронний ресурс] // Режим доступу: https://habr.com/ru/post/106302/
12. ГОСТ 34.603-92 Межгосударственный стандарт. Информационная технология. Виды испытаний автоматизированых систем.
13. ГОСТ РД 50-34.698-90 Автоматизированные системы. Требования к содержанию документов.

ДОДАТОК А

Тексти програмного коду

Інформаційна система удосконалення сприйняття інформації технології доповненої реальності

(Найменування програми (документа))

DVD-R

(Вид носія даних)

800 MB, 200 арк.

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

Adapter.java

```
package com.shvets.dp;
import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.List;
public class Adapter extends RecyclerView.Adapter<Adapter.ViewHolder>{
    private LayoutInflater layoutInflater;
    //private List<String> titles;
    private List<Integer> images;
    //int images[];
    private List<Model> mModels;
    private OnModelClickListener mOnModelClickListener;
    Adapter(Context context, List<Model> models, OnModelClickListener
onModelClickListener){
        this.layoutInflater = LayoutInflater.from(context);
        this.mModels = models;
        this.mOnModelClickListener = onModelClickListener;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
```

```
        View view = layoutInflater.inflate(R.layout.card_item_for_rv_models,parent,false);
        return new ViewHolder(view, mOnModelClickListener);
    }
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        //holder.image.setImageResource(images.get(position));
        holder.title.setText(mModels.get(position).getTitle());
        holder.image.setImageResource(mModels.get(position).getImageId());
    }
    @Override
    public int getItemCount() {
        //return titles.size();
        return mModels.size();
    }
    public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {

        TextView title;
        ImageView image;
        OnModelClickListener onModelClickListener;
        public ViewHolder(@NonNull View itemView, OnModelClickListener onModelClickListener) {
            super(itemView);
            title = itemView.findViewById(R.id.text_view_title_for_model);
            image = itemView.findViewById(R.id.image_view_for_online_courses);
            this.onModelClickListener = onModelClickListener;
            itemView.setOnClickListener(this);
        }
```

```
        @Override
        public void onClick(View v) {
            onModelClickListener.onModelClick(getAdapterPosition());
        }
    }
    public interface OnModelClickListener{
        void onModelClick(int position);
    }
}
```

Gltf.java

```
package com.shvets.dp;
import static java.util.concurrent.TimeUnit.SECONDS;
import android.app.Activity;
import android.app.ActivityManager;
import android.content.Context;
import android.net.Uri;
import android.os.Build;
import android.os.Build.VERSION_CODES;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.ArraySet;
import android.util.Log;
import android.view.Gravity;
import android.view.MotionEvent;
import android.widget.Toast;
import com.google.android.filament.gltfio.Animator;
import com.google.android.filament.gltfio.FilamentAsset;
import com.google.ar.core.Anchor;
import com.google.ar.core.HitResult;
```

```
import com.google.ar.core.Plane;
import com.google.ar.sceneform.AnchorNode;
import com.google.ar.sceneform.rendering.Color;
import com.google.ar.sceneform.rendering.Material;
import com.google.ar.sceneform.rendering.ModelRenderable;
import com.google.ar.sceneform.rendering.Renderable;
import com.google.ar.sceneform.ux.ArFragment;
import com.google.ar.sceneform.ux.TransformableNode;
import java.lang.ref.WeakReference;
import java.util.Arrays;
import java.util.List;
import java.util.Set;
public class GltfActivity extends AppCompatActivity {
  private static final String TAG = GltfActivity.class.getSimpleName();
  private static final double MIN_OPENGL_VERSION = 3.0;
  private ArFragment arFragment;
  private Renderable renderable;
  private static class AnimationInstance {
    Animator animator;
    Long startTime;
    float duration;
    int index;
    AnimationInstance(Animator animator, int index, Long startTime) {
      this.animator = animator;
      this.startTime = startTime;
      this.duration = animator.getAnimationDuration(index);
      this.index = index;
    }
  }
```

```
  private final Set<AnimationInstance> animators = new ArraySet<>();
  private final List<Color> colors =
      Arrays.asList(
          new Color(0, 0, 0, 1),
          new Color(1, 0, 0, 1),
          new Color(0, 1, 0, 1),
          new Color(0, 0, 1, 1),
          new Color(1, 1, 0, 1),
          new Color(0, 1, 1, 1),
          new Color(1, 0, 1, 1),
          new Color(1, 1, 1, 1));
  private int nextColor = 0;
  @Override
  @SuppressWarnings({"AndroidApiChecker", "FutureReturnValueIgnored"})
  // CompletableFuture requires api level 24
  // FutureReturnValueIgnored is not valid
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (!checkIsSupportedDeviceOrFinish(this)) {
      return;
    }
    setContentView(R.layout.activity_ux);
    arFragment                    =                    (ArFragment)
getSupportFragmentManager().findFragmentById(R.id.ux_fragment);
    WeakReference<GltfActivity> weakActivity = new WeakReference<>(this);
    ModelRenderable.builder()
        .setSource(
            this,
            Uri.parse(
```

```
            "https://storage.googleapis.com/ar-answers-in-search-
models/static/Tiger/model.glb"))
        .setIsFilamentGltf(true)
        .build()
        .thenAccept(
            modelRenderable -> {
              GltfActivity activity = weakActivity.get();
              if (activity != null) {
                activity.renderable = modelRenderable;
              }
            })
        .exceptionally(
            throwable -> {
              Toast toast =
                  Toast.makeText(this, "Unable to load Tiger renderable",
Toast.LENGTH_LONG);
              toast.setGravity(Gravity.CENTER, 0, 0);
              toast.show();
              return null;
            });
    arFragment.setOnTapArPlaneListener(
        (HitResult hitResult, Plane plane, MotionEvent motionEvent) -> {
          if (renderable == null) {
            return;
          }
          // Create the Anchor.
          Anchor anchor = hitResult.createAnchor();
          AnchorNode anchorNode = new AnchorNode(anchor);
          anchorNode.setParent(arFragment.getArSceneView().getScene());
```

```
        // Create the transformable model and add it to the ancho
r.
        TransformableNode model = new
TransformableNode(arFragment.getTransformationSystem());
        model.setParent(anchorNode);
        model.setRenderable(renderable);
        model.select();
        FilamentAsset filamentAsset =
model.getRenderableInstance().getFilamentAsset();
        if (filamentAsset.getAnimator().getAnimationCount() > 0) {
          animators.add(new AnimationInstance(filamentAsset.getAnimator(), 0,
System.nanoTime()));
        }
        Color color = colors.get(nextColor);
        nextColor++;
        for (int i = 0; i < renderable.getSubmeshCount(); ++i) {
          Material material = renderable.getMaterial(i);
          material.setFloat4("baseColorFactor", color);
        }
      });
    arFragment
        .getArSceneView()
        .getScene()
        .addOnUpdateListener(
            frameTime -> {
              Long time = System.nanoTime();
              for (AnimationInstance animator : animators) {
                animator.animator.applyAnimation(
                    animator.index,
```

```
                (float) ((time - animator.startTime) / (double) SECONDS.toNanos(1))
                    % animator.duration);
            animator.animator.updateBoneMatrices();
          }
        });
  }
   * Returns false and displays an error message if Sceneform can not run, true if Sceneform can run
   * on this device.
   *
   * <p>Sceneform requires Android N on the device as well as OpenGL 3.0 capabilities.
   *
   * <p>Finishes the activity if Sceneform can not run
   */
  public static boolean checkIsSupportedDeviceOrFinish(final Activity activity) {
    if (Build.VERSION.SDK_INT < VERSION_CODES.N) {
      Log.e(TAG, "Sceneform requires Android N or later");
      Toast.makeText(activity, "Sceneform requires Android N or later", Toast.LENGTH_LONG).show();
      activity.finish();
      return false;
    }
    String openGlVersionString =
        ((ActivityManager)
activity.getSystemService(Context.ACTIVITY_SERVICE))
            .getDeviceConfigurationInfo()
            .getGlEsVersion();
    if (Double.parseDouble(openGlVersionString) < MIN_OPENGL_VERSION) {
```

```
      Log.e(TAG, "Sceneform requires OpenGL ES 3.0 later");
      Toast.makeText(activity, "Sceneform requires OpenGL ES 3.0 or later",
Toast.LENGTH_LONG)
          .show();
      activity.finish();
      return false;
    }
    return true;
  }
}
```

LoginActivity.java

```
package com.shvets.dp;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.ViewStub;
public class LoginActivity extends AppCompatActivity implements
View.OnClickListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        findViewById(R.id.cirLoginButton).setOnClickListener(this);
        /*ViewStub stub = (ViewStub) findViewById(R.id.layout_stub);
        stub.setLayoutResource(R.layout.layout_login);
        View inflated = stub.inflate();
*/
    }
```

```
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.cirLoginButton:
                Intent intent = new Intent(this, ModelsActivity.class);
                startActivity(intent);
                break;
            case R.id.textView2:
                break;
        }
    }
}
```

Model.java

```
package com.shvets.dp;
import java.util.ArrayList;
public class Model {
    private String mTitle;
    private int mImageId;
    public Model(String title, int imageId){
        mTitle = title;
        mImageId = imageId;
    }
    public String getTitle(){
        return mTitle;
    }
    public int getImageId(){
        return mImageId;
    }
    private static int lastModelId = 0;
```

```
    public static ArrayList<Model> modelArrayList(){
        ArrayList<Model> models = new ArrayList<>();
        models.add(new Model("Cactus", R.drawable.cactus));
        models.add(new Model("Buster drone", R.drawable.drone));
        models.add(new Model("Macbook", R.drawable.mac));
        models.add(new Model("Tigr", R.drawable.tigr));
        models.add(new Model("Skull", R.drawable.skull));
        models.add(new Model("Weapon", R.drawable.enemy));
        models.add(new Model("Penguin", R.drawable.penguin));
        return models;
    }
}
```

ModelDetailsActivity.java

```
package com.shvets.dp;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
public class ModelDetailsActivity extends AppCompatActivity implements
View.OnClickListener {
    ImageButton arBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_model_details);
        TextView textView = findViewById(R.id.tv_title_modelDetails);
```

```
            ImageView imageView = findViewById(R.id.main_backdrop);
            findViewById(R.id.fab_ar).setOnClickListener(this);
            Intent intent = getIntent();
            String title = intent.getStringExtra("selected_model_title");
            int image = intent.getIntExtra("selected_model_image",
R.drawable.ic_launcher_background);
            textView.setText(title);
            imageView.setImageResource(image);
        }

        @Override
        public void onClick(View view) {
            switch (view.getId()){
                case R.id.fab_ar:
                    Intent intent = new Intent(this, GltfActivity.class);
                    startActivity(intent);
                    break;
            }
        }
    }
    package com.shvets.dp;
    import android.content.Intent;
    import android.os.Bundle;
    import android.support.v7.app.AppCompatActivity;
    import android.support.v7.widget.GridLayoutManager;
    import android.support.v7.widget.RecyclerView;
    import java.util.ArrayList;
    public class ModelsActivity extends AppCompatActivity implements
Adapter.OnModelClickListener {
```

```
        RecyclerView recyclerView;
        public Adapter adapter;
        ArrayList<Model> modelArrayList;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_models);
            modelArrayList = Model.modelArrayList();
            recyclerView = findViewById(R.id.rv_models);
            recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
            adapter = new Adapter(this, modelArrayList, this);
            recyclerView.setAdapter(adapter);
        }
        @Override
        public void onModelClick(int position) {
            Intent intent = new Intent(this, ModelDetailsActivity.class);
            intent.putExtra("selected_model_title",
modelArrayList.get(position).getTitle());
            intent.putExtra("selected_model_image",
modelArrayList.get(position).getImageId());
            startActivity(intent);
        }
    }
    package org.tensorflow.lite.examples.classification;
    import android.Manifest;
    import android.app.Fragment;
    import android.content.Context;
    import android.content.pm.PackageManager;
    import android.hardware.Camera;
```

```
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.Image;
import android.media.Image.Plane;
import android.media.ImageReader;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.Trace;
import androidx.annotation.NonNull;
import androidx.annotation.UiThread;
import androidx.appcompat.app.AppCompatActivity;
import android.util.Size;
import android.view.Surface;
import android.view.View;
import android.view.ViewTreeObserver;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.material.bottomsheet.BottomSheetBehavior;
import java.nio.ByteBuffer;
```

```
import java.util.List;
import org.tensorflow.lite.examples.classification.env.ImageUtils;
import org.tensorflow.lite.examples.classification.env.Logger;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Device;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Model;
import
org.tensorflow.lite.examples.classification.tflite.Classifier.Recognition;
public abstract class CameraActivity extends AppCompatActivity
  implements OnImageAvailableListener,
    Camera.PreviewCallback,
    View.OnClickListener,
    AdapterView.OnItemSelectedListener {
  private static final Logger LOGGER = new Logger();
  private static final int PERMISSIONS_REQUEST = 1;
  private static final String PERMISSION_CAMERA =
Manifest.permission.CAMERA;
  protected int previewWidth = 0;
  protected int previewHeight = 0;
  private Handler handler;
  private HandlerThread handlerThread;
  private boolean useCamera2API;
  private boolean isProcessingFrame = false;
  private byte[][] yuvBytes = new byte[3][];
  private int[] rgbBytes = null;
  private int yRowStride;
  private Runnable postInferenceCallback;
  private Runnable imageConverter;
  private LinearLayout bottomSheetLayout;
  private LinearLayout gestureLayout;
```

```
    private BottomSheetBehavior<LinearLayout> sheetBehavior;
    protected TextView recognitionTextView,
        recognition1TextView,
        recognition2TextView,
        recognitionValueTextView,
        recognition1ValueTextView,
        recognition2ValueTextView;
    protected TextView frameValueTextView,
        cropValueTextView,
        cameraResolutionTextView,
        rotationTextView,
        inferenceTimeTextView;
    protected ImageView bottomSheetArrowImageView;
    private ImageView plusImageView, minusImageView;
    private Spinner modelSpinner;
    private Spinner deviceSpinner;
    private TextView threadsTextView;
    private Model model = Model.QUANTIZED_EFFICIENTNET;
    private Device device = Device.CPU;
    private int numThreads = -1;
    @Override
    protected void onCreate(final Bundle savedInstanceState) {
      LOGGER.d("onCreate " + this);
      super.onCreate(null);

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_
ON);
      setContentView(R.layout.tfe_ic_activity_camera);
      if (hasPermission()) {
```

```
            setFragment();
          } else {
            requestPermission();
          }
          threadsTextView = findViewById(R.id.threads);
          plusImageView = findViewById(R.id.plus);
          minusImageView = findViewById(R.id.minus);
          modelSpinner = findViewById(R.id.model_spinner);
          deviceSpinner = findViewById(R.id.device_spinner);
          bottomSheetLayout = findViewById(R.id.bottom_sheet_layout);
          gestureLayout = findViewById(R.id.gesture_layout);
          sheetBehavior = BottomSheetBehavior.from(bottomSheetLayout);
          bottomSheetArrowImageView =
findViewById(R.id.bottom_sheet_arrow);
          ViewTreeObserver vto = gestureLayout.getViewTreeObserver();
          vto.addOnGlobalLayoutListener(
              new ViewTreeObserver.OnGlobalLayoutListener() {
                @Override
                public void onGlobalLayout() {
                  if (Build.VERSION.SDK_INT <
Build.VERSION_CODES.JELLY_BEAN) {

gestureLayout.getViewTreeObserver().removeGlobalOnLayoutListener(this);
                  } else
{gestureLayout.getViewTreeObserver().removeOnGlobalLayoutListener(this);
                  }
                  int height = gestureLayout.getMeasuredHeight();
                  sheetBehavior.setPeekHeight(height);
                }
```

```
            });
        sheetBehavior.setHideable(false);
        sheetBehavior.setBottomSheetCallback(
            new BottomSheetBehavior.BottomSheetCallback() {
              @Override
              public void onStateChanged(@NonNull View bottomSheet, int
newState) {
                switch (newState) {
                  case BottomSheetBehavior.STATE_HIDDEN:
                    break;
                  case BottomSheetBehavior.STATE_EXPANDED:
                    {

bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_down);
                    }
                    break;
                  case BottomSheetBehavior.STATE_COLLAPSED:
                    {

bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_up);
                    }
                    break;
                  case BottomSheetBehavior.STATE_DRAGGING:
                    break;
                  case BottomSheetBehavior.STATE_SETTLING:

bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_up);
                    break;
                }
```

```
                }
                @Override
                public void onSlide(@NonNull View bottomSheet, float slideOffset)
{}
              });
          recognitionTextView = findViewById(R.id.detected_item);
          recognitionValueTextView = findViewById(R.id.detected_item_value);
          recognition1TextView = findViewById(R.id.detected_item1);
          recognition1ValueTextView                                          =
findViewById(R.id.detected_item1_value);
          recognition2TextView = findViewById(R.id.detected_item2);
          recognition2ValueTextView                                          =
findViewById(R.id.detected_item2_value);
          frameValueTextView = findViewById(R.id.frame_info);
          cropValueTextView = findViewById(R.id.crop_info);
          cameraResolutionTextView = findViewById(R.id.view_info);
          rotationTextView = findViewById(R.id.rotation_info);
          inferenceTimeTextView = findViewById(R.id.inference_info);
          modelSpinner.setOnItemSelectedListener(this);
          deviceSpinner.setOnItemSelectedListener(this);
          plusImageView.setOnClickListener(this);
          minusImageView.setOnClickListener(this);
          model                                                              =
Model.valueOf(modelSpinner.getSelectedItem().toString().toUpperCase());
          device = Device.valueOf(deviceSpinner.getSelectedItem().toString());
          numThreads                                                         =
Integer.parseInt(threadsTextView.getText().toString().trim());
        }
        protected int[] getRgbBytes() {
```

```
        imageConverter.run();
        return rgbBytes;
      }
      protected int getLuminanceStride() {
        return yRowStride;
      }
      protected byte[] getLuminance() {
        return yuvBytes[0];
      }
      @Override
      public void onPreviewFrame(final byte[] bytes, final Camera camera) {
        if (isProcessingFrame) {
          LOGGER.w("Dropping frame!");
          return;
        }
        try {
          // Initialize the storage bitmaps once when the resolution is known.
          if (rgbBytes == null) {
            Camera.Size previewSize = camera.getParameters().getPreviewSize();
            previewHeight = previewSize.height;
            previewWidth = previewSize.width;
            rgbBytes = new int[previewWidth * previewHeight];
            onPreviewSizeChosen(new                    Size(previewSize.width,
previewSize.height), 90);
          }
        } catch (final Exception e) {
          LOGGER.e(e, "Exception!");
          return;
        }
```

```
            isProcessingFrame = true;
            yuvBytes[0] = bytes;
            yRowStride = previewWidth;
            imageConverter =
                new Runnable() {
                  @Override
                  public void run() {
                    ImageUtils.convertYUV420SPToARGB8888(bytes,  previewWidth,
previewHeight, rgbBytes);
                  }
                };
            postInferenceCallback =
                new Runnable() {
                  @Override
                  public void run() {
                    camera.addCallbackBuffer(bytes);
                    isProcessingFrame = false;
                  }
                };
            processImage();
          }
          @Override
          public void onImageAvailable(final ImageReader reader) {
            // We need wait until we have some size from onPreviewSizeChosen
            if (previewWidth == 0 || previewHeight == 0) {
              return;
            }
            if (rgbBytes == null) {
              rgbBytes = new int[previewWidth * previewHeight];
```

```
}
try {
  final Image image = reader.acquireLatestImage();
  if (image == null) {
    return;
  }
  if (isProcessingFrame) {
    image.close();
    return;
  }
  isProcessingFrame = true;
  Trace.beginSection("imageAvailable");
  final Plane[] planes = image.getPlanes();
  fillBytes(planes, yuvBytes);
  yRowStride = planes[0].getRowStride();
  final int uvRowStride = planes[1].getRowStride();
  final int uvPixelStride = planes[1].getPixelStride();
  imageConverter =
      new Runnable() {
        @Override
        public void run() {
          ImageUtils.convertYUV420ToARGB8888(
              yuvBytes[0],
              yuvBytes[1],
              yuvBytes[2],
              previewWidth,
              previewHeight,
              yRowStride,
              uvRowStride,
```

```
                uvPixelStride,
                rgbBytes);
          }
        };
    postInferenceCallback =
        new Runnable() {
          @Override
          public void run() {
            image.close();
            isProcessingFrame = false;
          }
        };
    processImage();
  } catch (final Exception e) {
    LOGGER.e(e, "Exception!");
    Trace.endSection();
    return;
  }
  Trace.endSection();
}
@Override
public synchronized void onStart() {
  LOGGER.d("onStart " + this);
  super.onStart();
}
@Override
public synchronized void onResume() {
  LOGGER.d("onResume " + this);
  super.onResume();
```

```
  handlerThread = new HandlerThread("inference");
  handlerThread.start();
  handler = new Handler(handlerThread.getLooper());
 }
 @Override
 public synchronized void onPause() {
  LOGGER.d("onPause " + this);
  handlerThread.quitSafely();
  try {
   handlerThread.join();
   handlerThread = null;
   handler = null;
  } catch (final InterruptedException e) {
   LOGGER.e(e, "Exception!");
  }
  super.onPause();
 }
 @Override
 public synchronized void onStop() {
  LOGGER.d("onStop " + this);
  super.onStop();
 }
 @Override
 public synchronized void onDestroy() {
  LOGGER.d("onDestroy " + this);
  super.onDestroy();
 }
 protected synchronized void runInBackground(final Runnable r) {
  if (handler != null) {
```

```
            handler.post(r);
          }
        }
        @Override
        public void onRequestPermissionsResult(
            final int requestCode, final String[] permissions, final int[] grantResults)
{
          super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
          if (requestCode == PERMISSIONS_REQUEST) {
            if (allPermissionsGranted(grantResults)) {
              setFragment();
            } else {
              requestPermission();
            }
          }
        }
        private static boolean allPermissionsGranted(final int[] grantResults) {
          for (int result : grantResults) {
            if (result != PackageManager.PERMISSION_GRANTED) {
              return false;
            }
          }
          return true;
        }
        private boolean hasPermission() {
          if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            return checkSelfPermission(PERMISSION_CAMERA) ==
PackageManager.PERMISSION_GRANTED;
```

```
            } else {
              return true;
            }
          }
          private void requestPermission() {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
              if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA))
{
                Toast.makeText(
                    CameraActivity.this,
                    "Camera permission is required for this demo",
                    Toast.LENGTH_LONG)
                .show();
              }
              requestPermissions(new String[] {PERMISSION_CAMERA},
PERMISSIONS_REQUEST);
            }
          }
          private boolean isHardwareLevelSupported(
              CameraCharacteristics characteristics, int requiredLevel) {
            int deviceLevel =
characteristics.get(CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LE
VEL);
            if (deviceLevel ==
CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_LEGACY)
{
              return requiredLevel == deviceLevel;
            }
            return requiredLevel <= deviceLevel;
```

```
        }
        private String chooseCamera() {
         final CameraManager manager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);
         try {
          for (final String cameraId : manager.getCameraIdList()) {
           final CameraCharacteristics characteristics =
manager.getCameraCharacteristics(cameraId);
           final Integer facing =
characteristics.get(CameraCharacteristics.LENS_FACING);
           if (facing != null && facing ==
CameraCharacteristics.LENS_FACING_FRONT) {
            continue;
           }
           final StreamConfigurationMap map =

characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATIO
N_MAP);
           if (map == null) {
            continue;
           }
           useCamera2API =
              (facing == CameraCharacteristics.LENS_FACING_EXTERNAL)
                 || isHardwareLevelSupported(
                    characteristics,
CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_FULL);
           LOGGER.i("Camera API lv2?: %s", useCamera2API);
           return cameraId;
          }
```

```
      } catch (CameraAccessException e) {
        LOGGER.e(e, "Not allowed to access camera");
      }
      return null;
    }
    protected void setFragment() {
      String cameraId = chooseCamera();
      Fragment fragment;
      if (useCamera2API) {
        CameraConnectionFragment camera2Fragment =
            CameraConnectionFragment.newInstance(
                new CameraConnectionFragment.ConnectionCallback() {
                  @Override
                  public void onPreviewSizeChosen(final Size size, final int
rotation) {
                    previewHeight = size.getHeight();
                    previewWidth = size.getWidth();
                    CameraActivity.this.onPreviewSizeChosen(size, rotation);
                  }
                },
                this,
                getLayoutId(),
                getDesiredPreviewFrameSize());
        camera2Fragment.setCamera(cameraId);
        fragment = camera2Fragment;
      } else {
        fragment =
            new LegacyCameraConnectionFragment(this, getLayoutId(),
getDesiredPreviewFrameSize());
```

```
    }
    getFragmentManager().beginTransaction().replace(R.id.container,
fragment).commit();
  }
  protected void fillBytes(final Plane[] planes, final byte[][] yuvBytes) {
    // Because of the variable row stride it's not possible to know in
    // advance the actual necessary dimensions of the yuv planes.
    for (int i = 0; i < planes.length; ++i) {
      final ByteBuffer buffer = planes[i].getBuffer();
      if (yuvBytes[i] == null) {
        LOGGER.d("Initializing buffer %d at size %d", i, buffer.capacity());
        yuvBytes[i] = new byte[buffer.capacity()];
      }
      buffer.get(yuvBytes[i]);
    }
  }
  protected void readyForNextImage() {
    if (postInferenceCallback != null) {
      postInferenceCallback.run();
    }
  }
  protected int getScreenOrientation() {
    switch (getWindowManager().getDefaultDisplay().getRotation()) {
      case Surface.ROTATION_270:
        return 270;
      case Surface.ROTATION_180:
        return 180;
      case Surface.ROTATION_90:
        return 90;
```

```
            default:
              return 0;
            }
          }
          @UiThread
          protected void showResultsInBottomSheet(List<Recognition> results) {
            if (results != null && results.size() >= 3) {
              Recognition recognition = results.get(0);
              if (recognition != null) {
                if (recognition.getTitle() != null)
recognitionTextView.setText(recognition.getTitle());
                if (recognition.getConfidence() != null)
                  recognitionValueTextView.setText(
                      String.format("%.2f", (100 * recognition.getConfidence())) + "%");
              }
              Recognition recognition1 = results.get(1);
              if (recognition1 != null) {
                if (recognition1.getTitle() != null)
recognition1TextView.setText(recognition1.getTitle());
                if (recognition1.getConfidence() != null)
                  recognition1ValueTextView.setText(
                      String.format("%.2f", (100 * recognition1.getConfidence())) +
"%");
              }
              Recognition recognition2 = results.get(2);
              if (recognition2 != null) {
                if (recognition2.getTitle() != null)
recognition2TextView.setText(recognition2.getTitle());
                if (recognition2.getConfidence() != null)
```

```
                recognition2ValueTextView.setText(
                    String.format("%.2f", (100 * recognition2.getConfidence())) +
"%");
              }
            }
          }
          protected void showFrameInfo(String frameInfo) {
            frameValueTextView.setText(frameInfo);
          }
          protected void showCropInfo(String cropInfo) {
            cropValueTextView.setText(cropInfo);
          }
          protected void showCameraResolution(String cameraInfo) {
            cameraResolutionTextView.setText(cameraInfo);
          }
          protected void showRotationInfo(String rotation) {
            rotationTextView.setText(rotation);
          }
          protected void showInference(String inferenceTime) {
            inferenceTimeTextView.setText(inferenceTime);
          }
          protected Model getModel() {
            return model;
          }
          private void setModel(Model model) {
            if (this.model != model) {
              LOGGER.d("Updating  model: " + model);
              this.model = model;
              onInferenceConfigurationChanged();
```

```
    }
  }
  protected Device getDevice() {
    return device;
  }
  private void setDevice(Device device) {
    if (this.device != device) {
      LOGGER.d("Updating  device: " + device);
      this.device = device;
      final boolean threadsEnabled = device == Device.CPU;
      plusImageView.setEnabled(threadsEnabled);
      minusImageView.setEnabled(threadsEnabled);
      threadsTextView.setText(threadsEnabled ? String.valueOf(numThreads)
: "N/A");
      onInferenceConfigurationChanged();
    }
  }
  protected int getNumThreads() {
    return numThreads;
  }
  private void setNumThreads(int numThreads) {
    if (this.numThreads != numThreads) {
      LOGGER.d("Updating  numThreads: " + numThreads);
      this.numThreads = numThreads;
      onInferenceConfigurationChanged();
    }
  }
  protected abstract void processImage();
```

```
protected abstract void onPreviewSizeChosen(final Size size, final int rotation);
protected abstract int getLayoutId();
protected abstract Size getDesiredPreviewFrameSize();
protected abstract void onInferenceConfigurationChanged();
@Override
public void onClick(View v) {
  if (v.getId() == R.id.plus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads >= 9) return;
    setNumThreads(++numThreads);
    threadsTextView.setText(String.valueOf(numThreads));
  } else if (v.getId() == R.id.minus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads == 1) {
      return;
    }
    setNumThreads(--numThreads);
    threadsTextView.setText(String.valueOf(numThreads));
  }
}
@Override
public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
  if (parent == modelSpinner) {
```

```
setModel(Model.valueOf(parent.getItemAtPosition(pos).toString().toUpperCase())
);
          } else if (parent == deviceSpinner) {
            setDevice(Device.valueOf(parent.getItemAtPosition(pos).toString()));
          }
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
          // Do nothing.
        }
      }
      package org.tensorflow.lite.examples.classification;
      import android.annotation.SuppressLint;
      import android.app.Activity;
      import android.app.AlertDialog;
      import android.app.Dialog;
      import android.app.DialogFragment;
      import android.app.Fragment;
      import android.content.Context;
      import android.content.DialogInterface;
      import android.content.res.Configuration;
      import android.graphics.ImageFormat;
      import android.graphics.Matrix;
      import android.graphics.RectF;
      import android.graphics.SurfaceTexture;
      import android.hardware.camera2.CameraAccessException;
      import android.hardware.camera2.CameraCaptureSession;
      import android.hardware.camera2.CameraCharacteristics;
```

```
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.CaptureResult;
import android.hardware.camera2.TotalCaptureResult;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.ImageReader;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.text.TextUtils;
import android.util.Size;
import android.util.SparseIntArray;
import android.view.LayoutInflater;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.concurrent.Semaphore;
import java.util.concurrent.TimeUnit;
import
org.tensorflow.lite.examples.classification.customview.AutoFitTextureView;
```

```
import org.tensorflow.lite.examples.classification.env.Logger;

/**
 * Camera Connection Fragment that captures images from camera.
 *
 * <p>Instantiated by newInstance.</p>
 */
@SuppressWarnings("FragmentNotInstantiable")
public class CameraConnectionFragment extends Fragment {
  private static final Logger LOGGER = new Logger();
  /**
   * The camera preview size will be chosen to be the smallest frame by pixel size capable of
   * containing a DESIRED_SIZE x DESIRED_SIZE square.
   */
  private static final int MINIMUM_PREVIEW_SIZE = 320;
  /** Conversion from screen rotation to JPEG orientation. */
  private static final SparseIntArray ORIENTATIONS = new SparseIntArray();
  private static final String FRAGMENT_DIALOG = "dialog";
  static {
    ORIENTATIONS.append(Surface.ROTATION_0, 90);
    ORIENTATIONS.append(Surface.ROTATION_90, 0);
    ORIENTATIONS.append(Surface.ROTATION_180, 270);
    ORIENTATIONS.append(Surface.ROTATION_270, 180);
  }

  /** A {@link Semaphore} to prevent the app from exiting before closing the camera. */
```

```
    private final Semaphore cameraOpenCloseLock = new Semaphore(1);
    /** A {@link OnImageAvailableListener} to receive frames as they are
available. */
    private final OnImageAvailableListener imageListener;
    /** The input size in pixels desired by TensorFlow (width and height of a
square bitmap). */
    private final Size inputSize;
    /** The layout identifier to inflate for this Fragment. */
    private final int layout;
    private final ConnectionCallback cameraConnectionCallback;
    private final CameraCaptureSession.CaptureCallback captureCallback =
       new CameraCaptureSession.CaptureCallback() {
         @Override
         public void onCaptureProgressed(
             final CameraCaptureSession session,
             final CaptureRequest request,
             final CaptureResult partialResult) {}
         @Override
         public void onCaptureCompleted(
             final CameraCaptureSession session,
             final CaptureRequest request,
             final TotalCaptureResult result) {}
       };
    /** ID of the current {@link CameraDevice}. */
    private String cameraId;
    /** An {@link AutoFitTextureView} for camera preview. */
    private AutoFitTextureView textureView;
    /** A {@link CameraCaptureSession } for camera preview. */
    private CameraCaptureSession captureSession;
```

```
        /** A reference to the opened {@link CameraDevice}. */
        private CameraDevice cameraDevice;
        /** The rotation in degrees of the camera sensor from the display. */
        private Integer sensorOrientation;
        /** The {@link Size} of camera preview. */
        private Size previewSize;
        /** An additional thread for running tasks that shouldn't block the UI. */
        private HandlerThread backgroundThread;
        /** A {@link Handler} for running tasks in the background. */
        private Handler backgroundHandler;
        /**
         * {@link TextureView.SurfaceTextureListener} handles several lifecycle
events on a {@link
         * TextureView}.
         */
        private final TextureView.SurfaceTextureListener surfaceTextureListener
=
            new TextureView.SurfaceTextureListener() {
              @Override
              public void onSurfaceTextureAvailable(
                  final SurfaceTexture texture, final int width, final int height) {
               openCamera(width, height);
              }
              @Override
              public void onSurfaceTextureSizeChanged(
                  final SurfaceTexture texture, final int width, final int height) {
               configureTransform(width, height);
              }
              @Override
```

```
            public boolean onSurfaceTextureDestroyed(final SurfaceTexture
texture) {
              return true;
            }
            @Override
            public void onSurfaceTextureUpdated(final SurfaceTexture texture) {}
           };
        /** An {@link ImageReader} that handles preview frame capture. */
        private ImageReader previewReader;
        /** {@link CaptureRequest.Builder} for the camera preview */
        private CaptureRequest.Builder previewRequestBuilder;
        /** {@link CaptureRequest} generated by {@link
#previewRequestBuilder} */
        private CaptureRequest previewRequest;
        /** {@link CameraDevice.StateCallback} is called when {@link
CameraDevice} changes its state. */
        private final CameraDevice.StateCallback stateCallback =
          new CameraDevice.StateCallback() {
            @Override
            public void onOpened(final CameraDevice cd) {
              // This method is called when the camera is opened.  We start camera
preview here.
              cameraOpenCloseLock.release();
              cameraDevice = cd;
              createCameraPreviewSession();
            }
            @Override
            public void onDisconnected(final CameraDevice cd) {
              cameraOpenCloseLock.release();
```

```
            cd.close();
            cameraDevice = null;
          }
          @Override
          public void onError(final CameraDevice cd, final int error) {
            cameraOpenCloseLock.release();
            cd.close();
            cameraDevice = null;
            final Activity activity = getActivity();
            if (null != activity) {
              activity.finish();
            }
          }
        };
    @SuppressLint("ValidFragment")
    private CameraConnectionFragment(
        final ConnectionCallback connectionCallback,
        final OnImageAvailableListener imageListener,
        final int layout,
        final Size inputSize) {
      this.cameraConnectionCallback = connectionCallback;
      this.imageListener = imageListener;
      this.layout = layout;
      this.inputSize = inputSize;
    }
    /**
     * Given {@code choices} of {@code Size}s supported by a camera,
chooses the smallest one whose
```

```
     * width and height are at least as large as the minimum of both, or an exact
match if possible.
     *
     * @param choices The list of sizes that the camera supports for the
intended output class
     * @param width The minimum desired width
     * @param height The minimum desired height
     * @return The optimal {@code Size}, or an arbitrary one if none were big
enough
     */
    protected static Size chooseOptimalSize(final Size[] choices, final int
width, final int height) {
     final int minSize = Math.max(Math.min(width, height),
MINIMUM_PREVIEW_SIZE);
     final Size desiredSize = new Size(width, height);
     boolean exactSizeFound = false;
     final List<Size> bigEnough = new ArrayList<Size>();
     final List<Size> tooSmall = new ArrayList<Size>();
     for (final Size option : choices) {
      if (option.equals(desiredSize)) {
       exactSizeFound = true;
      }
      if (option.getHeight() >= minSize && option.getWidth() >= minSize) {
       bigEnough.add(option);
      } else {
       tooSmall.add(option);
      }
     }
```

```
        LOGGER.i("Desired size: " + desiredSize + ", min size: " + minSize + "x"
+ minSize);
        LOGGER.i("Valid preview sizes: [" + TextUtils.join(", ", bigEnough) +
"]");
        LOGGER.i("Rejected preview sizes: [" + TextUtils.join(", ", tooSmall) +
"]");
        if (exactSizeFound) {
          LOGGER.i("Exact size match found.");
          return desiredSize;
        }
        if (bigEnough.size() > 0) {
          final Size chosenSize = Collections.min(bigEnough, new
CompareSizesByArea());
          LOGGER.i("Chosen size: " + chosenSize.getWidth() + "x" +
chosenSize.getHeight());
          return chosenSize;
        } else {
          LOGGER.e("Couldn't find any suitable preview size");
          return choices[0];
        }
      }
      public static CameraConnectionFragment newInstance(
          final ConnectionCallback callback,
          final OnImageAvailableListener imageListener,
          final int layout,
          final Size inputSize) {
        return new CameraConnectionFragment(callback, imageListener, layout,
inputSize);
      }
```

```
    private void showToast(final String text) {
      final Activity activity = getActivity();
      if (activity != null) {
        activity.runOnUiThread(
            new Runnable() {
              @Override
              public void run() {
                Toast.makeText(activity, text, Toast.LENGTH_SHORT).show();
              }
            });
      }
    }
    @Override
    public View onCreateView(
        final LayoutInflater inflater, final ViewGroup container, final Bundle
savedInstanceState) {
      return inflater.inflate(layout, container, false);
    }
    @Override
    public void onViewCreated(final View view, final Bundle
savedInstanceState) {
      textureView = (AutoFitTextureView) view.findViewById(R.id.texture);
    }
    @Override
    public void onActivityCreated(final Bundle savedInstanceState) {
      super.onActivityCreated(savedInstanceState);
    }
    @Override
    public void onResume() {
```

```
        super.onResume();
        startBackgroundThread();
        if (textureView.isAvailable()) {
          openCamera(textureView.getWidth(), textureView.getHeight());
        } else {
          textureView.setSurfaceTextureListener(surfaceTextureListener);
        }
      }
      @Override
      public void onPause() {
        closeCamera();
        stopBackgroundThread();
        super.onPause();
      }
      public void setCamera(String cameraId) {
        this.cameraId = cameraId;
      }
      private void setUpCameraOutputs() {
        final Activity activity = getActivity();
        final CameraManager manager = (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
        try {
          final CameraCharacteristics characteristics =
manager.getCameraCharacteristics(cameraId);
          final StreamConfigurationMap map =

characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATIO
N_MAP);
```

```
            sensorOrientation =
characteristics.get(CameraCharacteristics.SENSOR_ORIENTATION);
            previewSize =
               chooseOptimalSize(
                  map.getOutputSizes(SurfaceTexture.class),
                  inputSize.getWidth(),
                  inputSize.getHeight());
            final int orientation = getResources().getConfiguration().orientation;
            if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
              textureView.setAspectRatio(previewSize.getWidth(),
previewSize.getHeight());
            } else {
              textureView.setAspectRatio(previewSize.getHeight(),
previewSize.getWidth());
            }
           } catch (final CameraAccessException e) {
            LOGGER.e(e, "Exception!");
           } catch (final NullPointerException e) {
            ErrorDialog.newInstance(getString(R.string.tfe_ic_camera_error))
               .show(getChildFragmentManager(), FRAGMENT_DIALOG);
            throw new
IllegalStateException(getString(R.string.tfe_ic_camera_error));
           }
           cameraConnectionCallback.onPreviewSizeChosen(previewSize,
sensorOrientation);
         }
         /** Opens the camera specified by {@link
CameraConnectionFragment#cameraId}. */
         private void openCamera(final int width, final int height) {
```

```
            setUpCameraOutputs();
            configureTransform(width, height);
            final Activity activity = getActivity();
            final CameraManager manager = (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
            try {
              if (!cameraOpenCloseLock.tryAcquire(2500,
TimeUnit.MILLISECONDS)) {
                throw new RuntimeException("Time out waiting to lock camera
opening.");
              }
              manager.openCamera(cameraId, stateCallback, backgroundHandler);
            } catch (final CameraAccessException e) {
              LOGGER.e(e, "Exception!");
            } catch (final InterruptedException e) {
              throw new RuntimeException("Interrupted while trying to lock camera
opening.", e);
            }
          }
          /** Closes the current {@link CameraDevice}. */
          private void closeCamera() {
            try {
              cameraOpenCloseLock.acquire();
              if (null != captureSession) {
                captureSession.close();
                captureSession = null;
              }
              if (null != cameraDevice) {
                cameraDevice.close();
```

```
            cameraDevice = null;
          }
          if (null != previewReader) {
            previewReader.close();
            previewReader = null;
          }
        } catch (final InterruptedException e) {
          throw new RuntimeException("Interrupted while trying to lock camera
closing.", e);
        } finally {
          cameraOpenCloseLock.release();
        }
      }
      /** Starts a background thread and its {@link Handler}. */
      private void startBackgroundThread() {
        backgroundThread = new HandlerThread("ImageListener");
        backgroundThread.start();
        backgroundHandler = new Handler(backgroundThread.getLooper());
      }
      /** Stops the background thread and its {@link Handler}. */
      private void stopBackgroundThread() {
        backgroundThread.quitSafely();
        try {
          backgroundThread.join();
          backgroundThread = null;
          backgroundHandler = null;
        } catch (final InterruptedException e) {
          LOGGER.e(e, "Exception!");
        }
```

```
        }
        private void createCameraPreviewSession() {
          try {
            final SurfaceTexture texture = textureView.getSurfaceTexture();
            assert texture != null;
            texture.setDefaultBufferSize(previewSize.getWidth(),
previewSize.getHeight());
            final Surface surface = new Surface(texture);
            previewRequestBuilder                                                =
cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
            previewRequestBuilder.addTarget(surface);
            LOGGER.i("Opening camera preview: " + previewSize.getWidth() + "x"
+ previewSize.getHeight());
            previewReader =
                ImageReader.newInstance(
                    previewSize.getWidth(),                previewSize.getHeight(),
ImageFormat.YUV_420_888, 2);
            previewReader.setOnImageAvailableListener(imageListener,
backgroundHandler);
            previewRequestBuilder.addTarget(previewReader.getSurface());
            cameraDevice.createCaptureSession(
                Arrays.asList(surface, previewReader.getSurface()),
                new CameraCaptureSession.StateCallback() {
                  @Override
                  public      void      onConfigured(final      CameraCaptureSession
cameraCaptureSession) {
                    if (null == cameraDevice) {
                      return;
                    }
```

```
                // When the session is ready, we start displaying the preview.
                captureSession = cameraCaptureSession;
                try {
                  // Auto focus should be continuous for camera preview.
                  previewRequestBuilder.set(
                      CaptureRequest.CONTROL_AF_MODE,

CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
                  // Flash is automatically enabled when necessary.
                  previewRequestBuilder.set(
                      CaptureRequest.CONTROL_AE_MODE,
CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);

                  // Finally, we start displaying the camera preview.
                  previewRequest = previewRequestBuilder.build();
                  captureSession.setRepeatingRequest(
                      previewRequest, captureCallback, backgroundHandler);
                } catch (final CameraAccessException e) {
                  LOGGER.e(e, "Exception!");
                }
              }
              @Override
              public void onConfigureFailed(final CameraCaptureSession
cameraCaptureSession) {
                showToast("Failed");
              }
            },
            null);
        } catch (final CameraAccessException e) {
```

```
            LOGGER.e(e, "Exception!");
          }
        }
        /**
         * Configures the necessary {@link Matrix} transformation to `mTextureView`. This method should be
         * called after the camera preview size is determined in setUpCameraOutputs and also the size of
         * `mTextureView` is fixed.
         *
         * @param viewWidth The width of `mTextureView`
         * @param viewHeight The height of `mTextureView`
         */
        private void configureTransform(final int viewWidth, final int viewHeight) {
          final Activity activity = getActivity();
          if (null == textureView || null == previewSize || null == activity) {
            return;
          }
          final int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
          final Matrix matrix = new Matrix();
          final RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
          final RectF bufferRect = new RectF(0, 0, previewSize.getHeight(), previewSize.getWidth());
          final float centerX = viewRect.centerX();
          final float centerY = viewRect.centerY();
          if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
```

```
            bufferRect.offset(centerX - bufferRect.centerX(), centerY -
bufferRect.centerY());
            matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
            final float scale =
                Math.max(
                    (float) viewHeight / previewSize.getHeight(),
                    (float) viewWidth / previewSize.getWidth());
            matrix.postScale(scale, scale, centerX, centerY);
            matrix.postRotate(90 * (rotation - 2), centerX, centerY);
           } else if (Surface.ROTATION_180 == rotation) {
            matrix.postRotate(180, centerX, centerY);
           }
           textureView.setTransform(matrix);
          }
          public interface ConnectionCallback {
           void onPreviewSizeChosen(Size size, int cameraRotation);
          }
          /** Compares two {@code Size}s based on their areas. */
          static class CompareSizesByArea implements Comparator<Size> {
           @Override
           public int compare(final Size lhs, final Size rhs) {
            // We cast here to ensure the multiplications won't overflow
            return Long.signum(
                (long) lhs.getWidth() * lhs.getHeight() - (long) rhs.getWidth() *
rhs.getHeight());
           }
          }
          /** Shows an error message dialog. */
          public static class ErrorDialog extends DialogFragment {
```

```
        private static final String ARG_MESSAGE = "message";

        public static ErrorDialog newInstance(final String message) {
          final ErrorDialog dialog = new ErrorDialog();
          final Bundle args = new Bundle();
          args.putString(ARG_MESSAGE, message);
          dialog.setArguments(args);
          return dialog;
        }
        @Override
        public Dialog onCreateDialog(final Bundle savedInstanceState) {
          final Activity activity = getActivity();
          return new AlertDialog.Builder(activity)
              .setMessage(getArguments().getString(ARG_MESSAGE))
              .setPositiveButton(
                  android.R.string.ok,
                  new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(final DialogInterface dialogInterface, final int
i) {
                      activity.finish();
                    }
                  })
              .create();
        }
      }
    }
    package org.tensorflow.lite.examples.classification;
    import android.graphics.Bitmap;
```

```
import android.graphics.Bitmap.Config;
import android.graphics.Typeface;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.SystemClock;
import android.util.Size;
import android.util.TypedValue;
import android.widget.Toast;
import java.io.IOException;
import java.util.List;
import org.tensorflow.lite.examples.classification.env.BorderedText;
import org.tensorflow.lite.examples.classification.env.Logger;
import org.tensorflow.lite.examples.classification.tflite.Classifier;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Device;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Model;
public class ClassifierActivity extends CameraActivity implements
OnImageAvailableListener {
  private static final Logger LOGGER = new Logger();
  private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);
  private static final float TEXT_SIZE_DIP = 10;
  private Bitmap rgbFrameBitmap = null;
  private long lastProcessingTimeMs;
  private Integer sensorOrientation;
  private Classifier classifier;
  private BorderedText borderedText;
  /** Input image size of the model along x axis. */
  private int imageSizeX;
  /** Input image size of the model along y axis. */
  private int imageSizeY;
  @Override
```

```
protected int getLayoutId() {
  return R.layout.tfe_ic_camera_connection_fragment;
}
@Override
protected Size getDesiredPreviewFrameSize() {
  return DESIRED_PREVIEW_SIZE;
}
@Override
public void onPreviewSizeChosen(final Size size, final int rotation) {
  final float textSizePx =
      TypedValue.applyDimension(
          TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
  borderedText = new BorderedText(textSizePx);
  borderedText.setTypeface(Typeface.MONOSPACE);
  recreateClassifier(getModel(), getDevice(), getNumThreads());
  if (classifier == null) {
   LOGGER.e("No classifier on preview!");
   return;
  }
  previewWidth = size.getWidth();
  previewHeight = size.getHeight();
  sensorOrientation = rotation - getScreenOrientation();
  LOGGER.i("Camera orientation relative to screen canvas: %d",
sensorOrientation);
  LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
  rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight,
Config.ARGB_8888);
}
```

```
        @Override
        protected void processImage() {
         rgbFrameBitmap.setPixels(getRgbBytes(), 0, previewWidth, 0, 0,
previewWidth, previewHeight);
         final int cropSize = Math.min(previewWidth, previewHeight);
         runInBackground(
            new Runnable() {
              @Override
              public void run() {
                if (classifier != null) {
                  final long startTime = SystemClock.uptimeMillis();
                  final List<Classifier.Recognition> results =
                     classifier.recognizeImage(rgbFrameBitmap, sensorOrientation);
                  lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;
                  LOGGER.v("Detect: %s", results);
                  runOnUiThread(
                     new Runnable() {
                       @Override
                       public void run() {
                         showResultsInBottomSheet(results);
                         showFrameInfo(previewWidth + "x" + previewHeight);
                         showCropInfo(imageSizeX + "x" + imageSizeY);
                         showCameraResolution(cropSize + "x" + cropSize);
                         showRotationInfo(String.valueOf(sensorOrientation));
                         showInference(lastProcessingTimeMs + "ms");
                       }
                     });
                }
                readyForNextImage();
```

```
            }
          });
      }
      @Override
      protected void onInferenceConfigurationChanged() {
        if (rgbFrameBitmap == null) {
          // Defer creation until we're getting camera frames.
          return;
        }
        final Device device = getDevice();
        final Model model = getModel();
        final int numThreads = getNumThreads();
        runInBackground(() -> recreateClassifier(model, device, numThreads));
      }
      private void recreateClassifier(Model model, Device device, int
numThreads) {
        if (classifier != null) {
          LOGGER.d("Closing classifier.");
          classifier.close();
          classifier = null;
        }
        if (device == Device.GPU
      && (model == Model.QUANTIZED_MOBILENET || model ==
Model.QUANTIZED_EFFICIENTNET)) {
          LOGGER.d("Not creating classifier: GPU doesn't support quantized
models.");
          runOnUiThread(
              () -> {
```

```
                Toast.makeText(this, R.string.tfe_ic_gpu_quant_error,
Toast.LENGTH_LONG).show();
              });
          return;
        }
        try {
          LOGGER.d(
              "Creating classifier (model=%s, device=%s, numThreads=%d)",
model, device, numThreads);
          classifier = Classifier.create(this, model, device, numThreads);
        } catch (IOException e) {
          LOGGER.e(e, "Failed to create classifier.");
        }
        // Updates the input image size.
        imageSizeX = classifier.getImageSizeX();
        imageSizeY = classifier.getImageSizeY();
      }
    }
    package org.tensorflow.lite.examples.classification;
    import android.annotation.SuppressLint;
    import android.app.Fragment;
    import android.graphics.SurfaceTexture;
    import android.hardware.Camera;
    import android.hardware.Camera.CameraInfo;
    import android.os.Bundle;
    import android.os.Handler;
    import android.os.HandlerThread;
    import android.util.Size;
    import android.util.SparseIntArray;
```

```
import android.view.LayoutInflater;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.view.ViewGroup;
import java.io.IOException;
import java.util.List;
import
org.tensorflow.lite.examples.classification.customview.AutoFitTextureView;
import org.tensorflow.lite.examples.classification.env.ImageUtils;
import org.tensorflow.lite.examples.classification.env.Logger;
public class LegacyCameraConnectionFragment extends Fragment {
  private static final Logger LOGGER = new Logger();
  /** Conversion from screen rotation to JPEG orientation. */
  private static final SparseIntArray ORIENTATIONS = new
SparseIntArray();
  static {
    ORIENTATIONS.append(Surface.ROTATION_0, 90);
    ORIENTATIONS.append(Surface.ROTATION_90, 0);
    ORIENTATIONS.append(Surface.ROTATION_180, 270);
    ORIENTATIONS.append(Surface.ROTATION_270, 180);
  }
  private Camera camera;
  private Camera.PreviewCallback imageListener;
  private Size desiredSize;
  /** The layout identifier to inflate for this Fragment. */
  private int layout;
  /** An {@link AutoFitTextureView} for camera preview. */
  private AutoFitTextureView textureView;
```

```
        /**
         * {@link TextureView.SurfaceTextureListener} handles several lifecycle
events on a {@link
         * TextureView}.
         */
        private final TextureView.SurfaceTextureListener surfaceTextureListener
=
            new TextureView.SurfaceTextureListener() {
              @Override
              public void onSurfaceTextureAvailable(
                  final SurfaceTexture texture, final int width, final int height) {
               int index = getCameraId();
               camera = Camera.open(index);
               try {
                 Camera.Parameters parameters = camera.getParameters();
                 List<String> focusModes = parameters.getSupportedFocusModes();
                 if (focusModes != null
       &&
focusModes.contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICT
URE)) {

parameters.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_
PICTURE);
                 }
                 List<Camera.Size>                cameraSizes                =
parameters.getSupportedPreviewSizes();
                 Size[] sizes = new Size[cameraSizes.size()];
                 int i = 0;
                 for (Camera.Size size : cameraSizes) {
```

```
                sizes[i++] = new Size(size.width, size.height);
              }
              Size previewSize =
                 CameraConnectionFragment.chooseOptimalSize(
                     sizes, desiredSize.getWidth(), desiredSize.getHeight());
              parameters.setPreviewSize(previewSize.getWidth(),
previewSize.getHeight());
               camera.setDisplayOrientation(90);
               camera.setParameters(parameters);
               camera.setPreviewTexture(texture);
              } catch (IOException exception) {
               camera.release();
              }
              camera.setPreviewCallbackWithBuffer(imageListener);
              Camera.Size s = camera.getParameters().getPreviewSize();
              camera.addCallbackBuffer(new
byte[ImageUtils.getYUVByteSize(s.height, s.width)]);
              textureView.setAspectRatio(s.height, s.width);
              camera.startPreview();
            }
            @Override
            public void onSurfaceTextureSizeChanged(
               final SurfaceTexture texture, final int width, final int height) {}
            @Override
            public boolean onSurfaceTextureDestroyed(final SurfaceTexture
texture) {
              return true;
            }
            @Override
```

```
            public void onSurfaceTextureUpdated(final SurfaceTexture texture) {}
          };
        private HandlerThread backgroundThread;
        @SuppressLint("ValidFragment")
        public LegacyCameraConnectionFragment(
          final Camera.PreviewCallback imageListener, final int layout, final Size
desiredSize) {
          this.imageListener = imageListener;
          this.layout = layout;
          this.desiredSize = desiredSize;
        }
        @Override
        public View onCreateView(
          final LayoutInflater inflater, final ViewGroup container, final Bundle
savedInstanceState) {
          return inflater.inflate(layout, container, false);
        }
        @Override
        public void onViewCreated(final View view, final Bundle
savedInstanceState) {
          textureView = (AutoFitTextureView) view.findViewById(R.id.texture);
        }
        @Override
        public void onActivityCreated(final Bundle savedInstanceState) {
          super.onActivityCreated(savedInstanceState);
        }
        @Override
        public void onResume() {
          super.onResume();
```

```
  startBackgroundThread();
  if (textureView.isAvailable()) {
    if (camera != null) {
      camera.startPreview();
    }
  } else {
    textureView.setSurfaceTextureListener(surfaceTextureListener);
  }
}
@Override
public void onPause() {
  stopCamera();
  stopBackgroundThread();
  super.onPause();
}
/** Starts a background thread and its {@link Handler}. */
private void startBackgroundThread() {
  backgroundThread = new HandlerThread("CameraBackground");
  backgroundThread.start();
}
/** Stops the background thread and its {@link Handler}. */
private void stopBackgroundThread() {
  backgroundThread.quitSafely();
  try {
    backgroundThread.join();
    backgroundThread = null;
  } catch (final InterruptedException e) {
    LOGGER.e(e, "Exception!");
  }
```

```
  }
  protected void stopCamera() {
    if (camera != null) {
      camera.stopPreview();
      camera.setPreviewCallback(null);
      camera.release();
      camera = null;
    }
  }
  private int getCameraId() {
    CameraInfo ci = new CameraInfo();
    for (int i = 0; i < Camera.getNumberOfCameras(); i++) {
      Camera.getCameraInfo(i, ci);
      if (ci.facing == CameraInfo.CAMERA_FACING_BACK) return i;
    }
    return -1; // No camera found
  }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТімені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО
Керівник проєкту
_____ *СвітланаПроскура*
(підпис) (вл. ім’я, прізвище)
“13” квітня2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)
“14” квітня2020 р.

ІНФОРМАЦІЙНА СИСТЕМА УДОСКОНАЛЕННЯ СПРИЙНЯТТЯ ІНФОРМАЦІЇ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6330.01.000 ТЗ*

на 10 сторінках

Київ – 2020 року

ЗМІСТ

Зм.	*Арк.*	*Прізвище*	*Підпис*	*Дата*			
Розроб.		*Швець Д.Ю.*			*Інформаційна системаудосконалення сприйняття інформації технології*	*Літ.*	*Лист* 2 / *Листів* 10
Перевірив.		*Проскура С.Л.*					
						КПІ ім. ІгоряСікорського Каф. АСОІУ Гр. ІС-63	
Н. кон.		*Проскура С.Л.*					
Затв.		*Павлов О.А.*					

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Повне найменування системи та її умовне позначення

Повна назва системи: «Інформаційна система удосконалення сприйняття інформації технології доповненої реальності».

Коротке найменування системи: «ARWorld».

1.2. Найменування організації-замовника та організацій-учасників робіт

Генеральним замовником проекту являється кафедра «Автоматизованих систем обробки інформації та управління» «КПІ ім. Ігоря Сікорьского». Представником замовника є Проскура Світлана Леонідівна.

Розробником системи є студентка групи ІС-63 факультету інформатики та обчислювальної техніки «КПІ ім. Ігоря Сікорського» Швець Дар'я Юріївна.

1.3. Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4. Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням системи – 5 лютого 2020 рік.

Плановий термін по закінченню роботи над створенням системи – не пізніше 15 червня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1. Призначення програмного продукту

Система призначена для того, щоб допомогти компаніям впроваджувати доповнену реальність на своїх підприємствах та полегшити її сприйняття користувачами.

2.2. Цілі створення системи

Основними цілями розробки інформаційної системи є:

- полегшення створення 3D-моделей;
- оптимізація етапу накладання 3D-моделей на об’єкти;
- полегшення управління доповненою інформацією для усіх товарів.

Для досягнення поставлених цілей мають бути вирішені такі задачі:

- створення 3D-моделей;
- аналіз реального світу через камеру смартфона;
- накладання моделі на певний об’єкт з навколишнього середовища;
- вивід товару з доповненою реальністю на ринок.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування сервісом обов’язковою умовою для користувача є наявність смартфона на базі операційної системи Android версії не нижче 7.0, адже даний програмний продукт є мобільним застосуванням з мінімальним рівнем API – 24.

Працювати з сервісом може як авторизований користувач, так і неавторизований, однак останні можуть не мати доступу до усього функціоналу системи. Після авторизації, користувач отримає доступ до додавання своїх моделей та предметів для сканування.

Користувач, що є компанією, має особливі можливості, такі як публікувати товари для можливості їх сканування користувачами.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система має здійснювати сканування предметів з доповненою реальністю та виводити додаткову інформацію, надавати можливість накладати AR на об’єкти з зображення камери смартфона. Система має виконувати наступні функції:

- робота з обліковими записами;
- сканування об’єктів, які опубліковані у мережу;
- публікування компаніями предметів з додатковою інформацією;
- накладання доповненої реальності на реальний світ;
- створювання нових 3D-моделей.

4.2 Вимоги до надійності

Програма повинна зберігати працездатність і забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

- при помилках в роботі апаратних засобів (крім носіїв даних і програм). Відновлення функції програми покладається на хостинг;
- при помилках, пов’язаних з особливістю різноманітних смартфонів, існуючих на сьогоднішній день.

Програмний продукт повинен поєднувати надійність та функціональність. У разі виникнення аварійних ситуацій необхідно сповіщати користувача та надавати інструкцію для подальших дій. Будь-які аварійні ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.3 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити виконання встановлених операцій процесу технічного обслуговування.

Для правильної роботи розробленої системи до складу технічних засобів повинен входити смартфон, що має конфігурацію наведену нижче:

- процесор з тактовою частотою не нижче 1,6 ГГц;
- об’єм оперативної пам’яті не менше 1024 МБ;
- об’єм пам’яті не нижчий за 2Гб;
- доступ до камери;
- доступ до мережі інтернет.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки програмного забезпечення наведені в таблиці 5.1.

Таблиця 5.1

№	Назва етапу	Термін виконання
1.	Вивчення рекомендованої літератури	01.03.2020
2.	Аналіз існуючих методів розв’язання задачі	07.03.2020
3.	Постановка та формалізація задачі	15.03.2020
4.	Розробка програмного забезпечення	03.05.2020
5.	Налагодження програми	07.05.2020
6.	Тестування програми	10.05.2020
7.	Оформлення пояснювальної записки	14.05.2020
8.	Подання ДП на попередній захист	15.05.2020
9.	Подання ДП на основний захист	15.06.2020

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Для перевірки контролю правильності роботи програмного забезпечення відбувається функціональне тестування. В ході тестування проведено випробування основних модулів системи. Список проведених тестувань:

- тестування сканування об’єктів;
- тестування накладання доповненої реальності на об’єкт;
- тестування створювання моделей доповненої реальності;
- тестування публікації товарів;
- тестування використання моделей з каталогу;
- тестування припинення інвестування проекту;
- тестування загрузки своїх моделей;
- тестування фільтрування моделей;
- тестування класифікації моделей;
- тестування авторизації в системі;
- тестування реєстрації в системі;
- тестування виходу із системи;
- тестування перегляду моделей і товарів;
- тестування пошуку моделей і товарів;
- тестування видалення моделей і товарів.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1004014820

Дата перевірки:
13.06.2020 02:31:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2020 18:14:32 EEST

ID користувача:
77149

Назва документу: **+Schvets_is63**

ID файлу: **1004027871** Кількість сторінок: **45** Кількість слів: **6041** Кількість символів: **49987** Розмір файлу: **641.56 KB**

16.9% Схожість

Найбільша схожість: **4.68%** з джерело бібліотеки. ID файлу: **1000018131**

5.86% Схожість з Інтернет джерелами	44	Page 47
16.1% Текстові збіги по Бібліотеці акаунту	257	Page 47

0% Цитат

Не знайдено жодних цитат

64.6% Вилучень

Джерела менше, ніж **8 слів** автоматично вилучено

Не знайдено жодного вилученого тексту з Інтернету

64.6% Вилученого тексту з Бібліотеки	1	Page 47

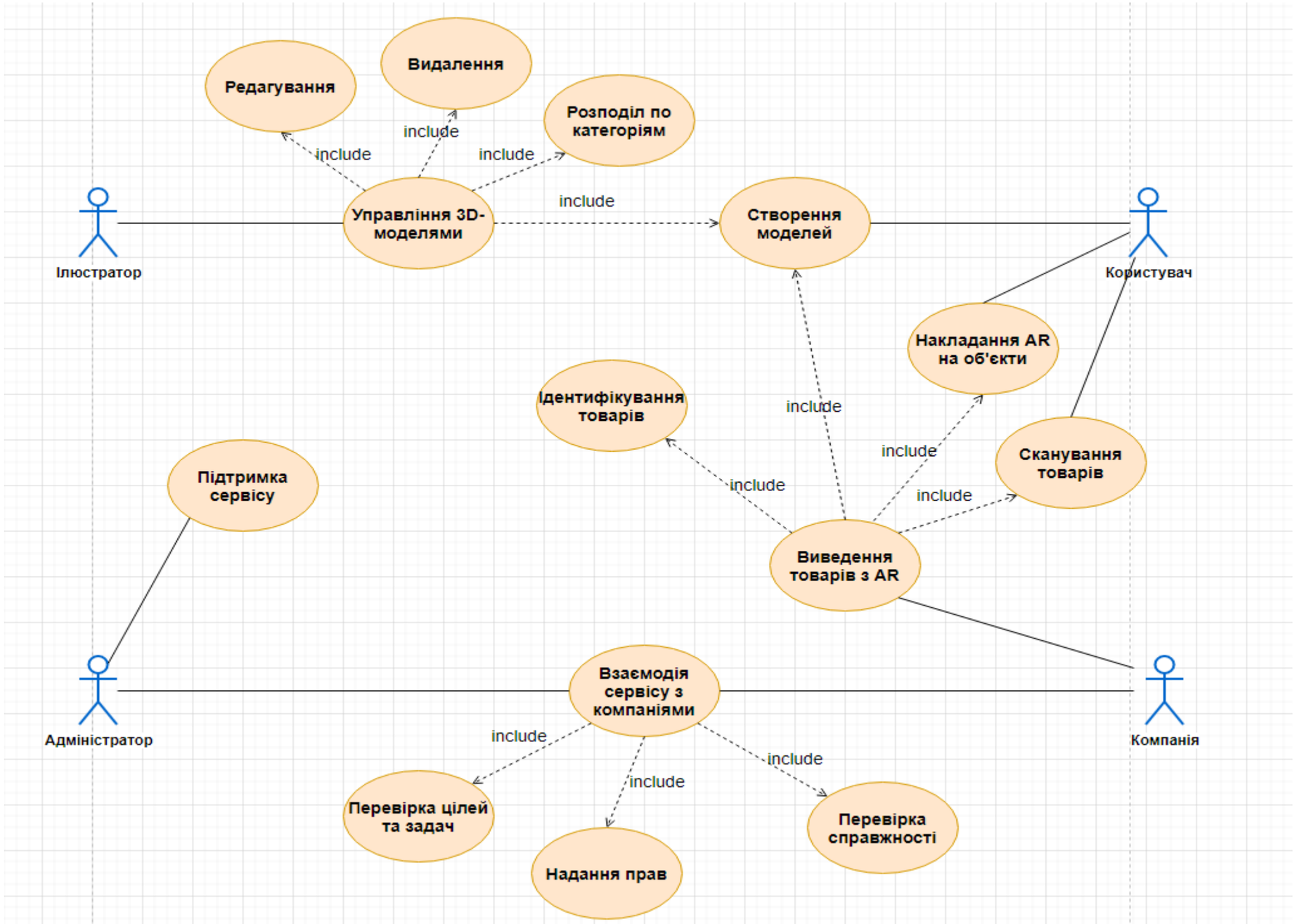
Підміна символів

Заміна символів	6

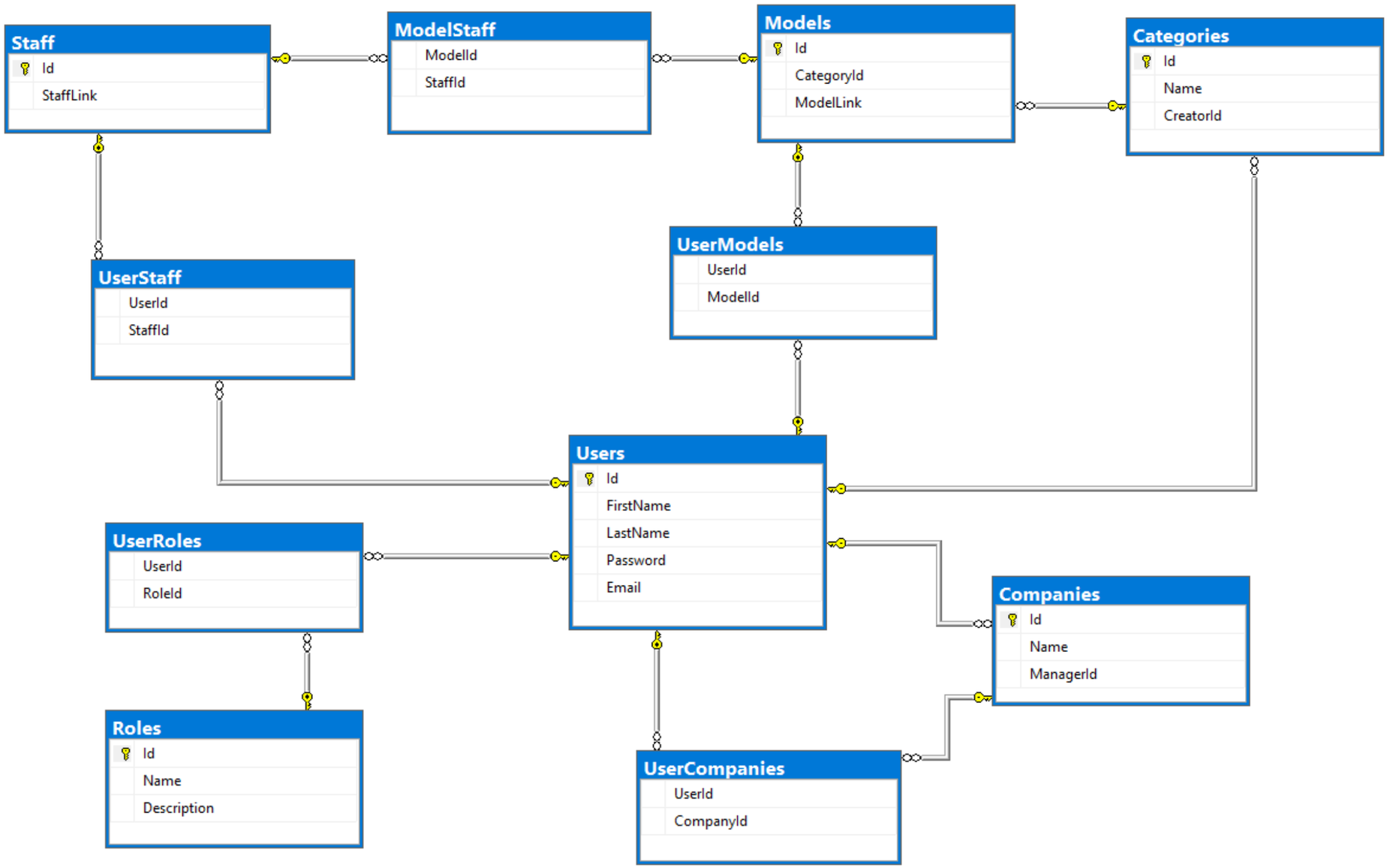
Графічний матеріал
до дипломного проєкту

на тему: Інформаційна система удосконалення сприйняття інформації технології доповненої реальності

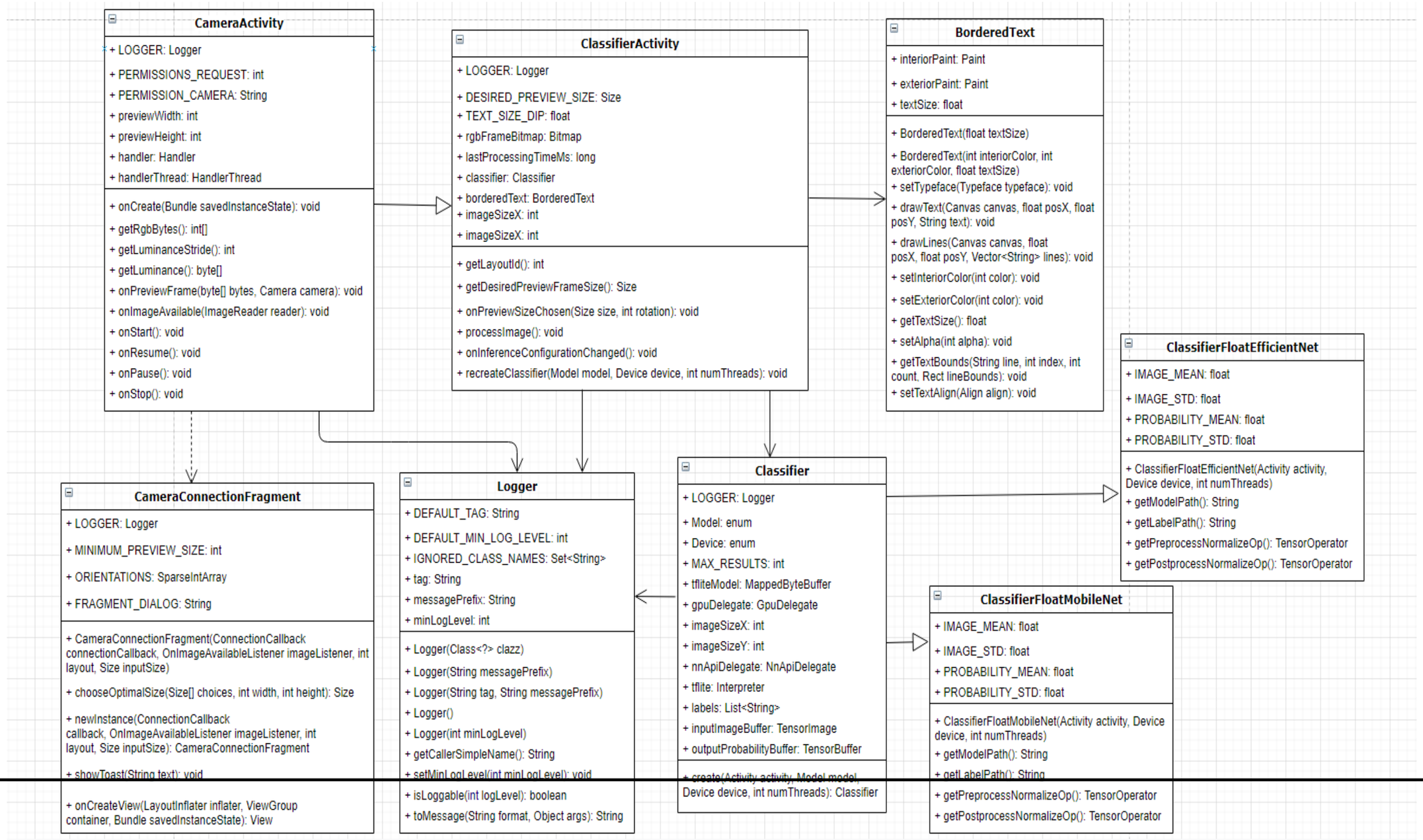
Київ – 2020 року

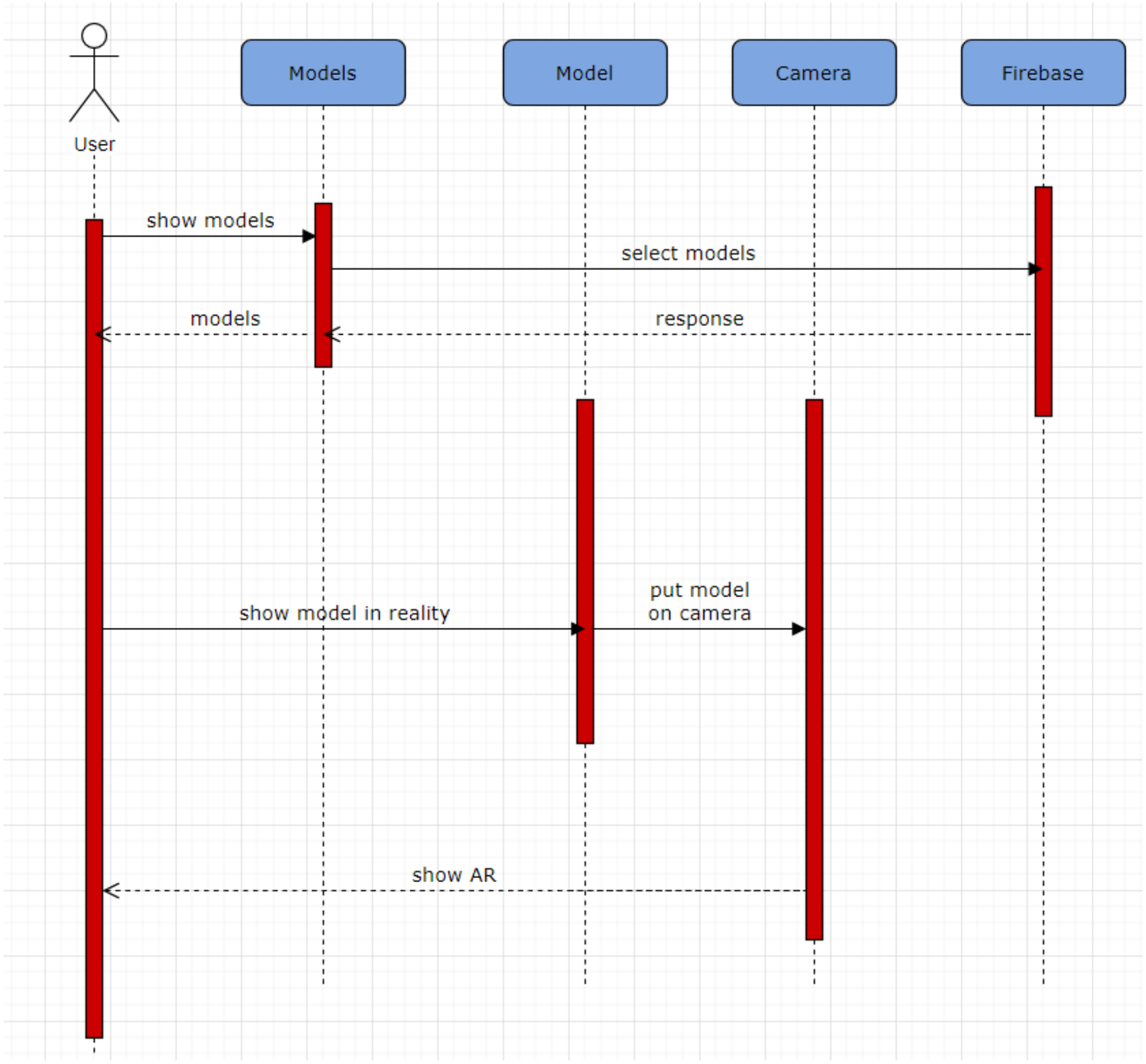


					ДП 6330.02.000 ССВ	
Розробн.					Схема структурна варіантів використань	
Керівн.						
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63
		Проскура С.Л.				
		Проскура С.Л.				

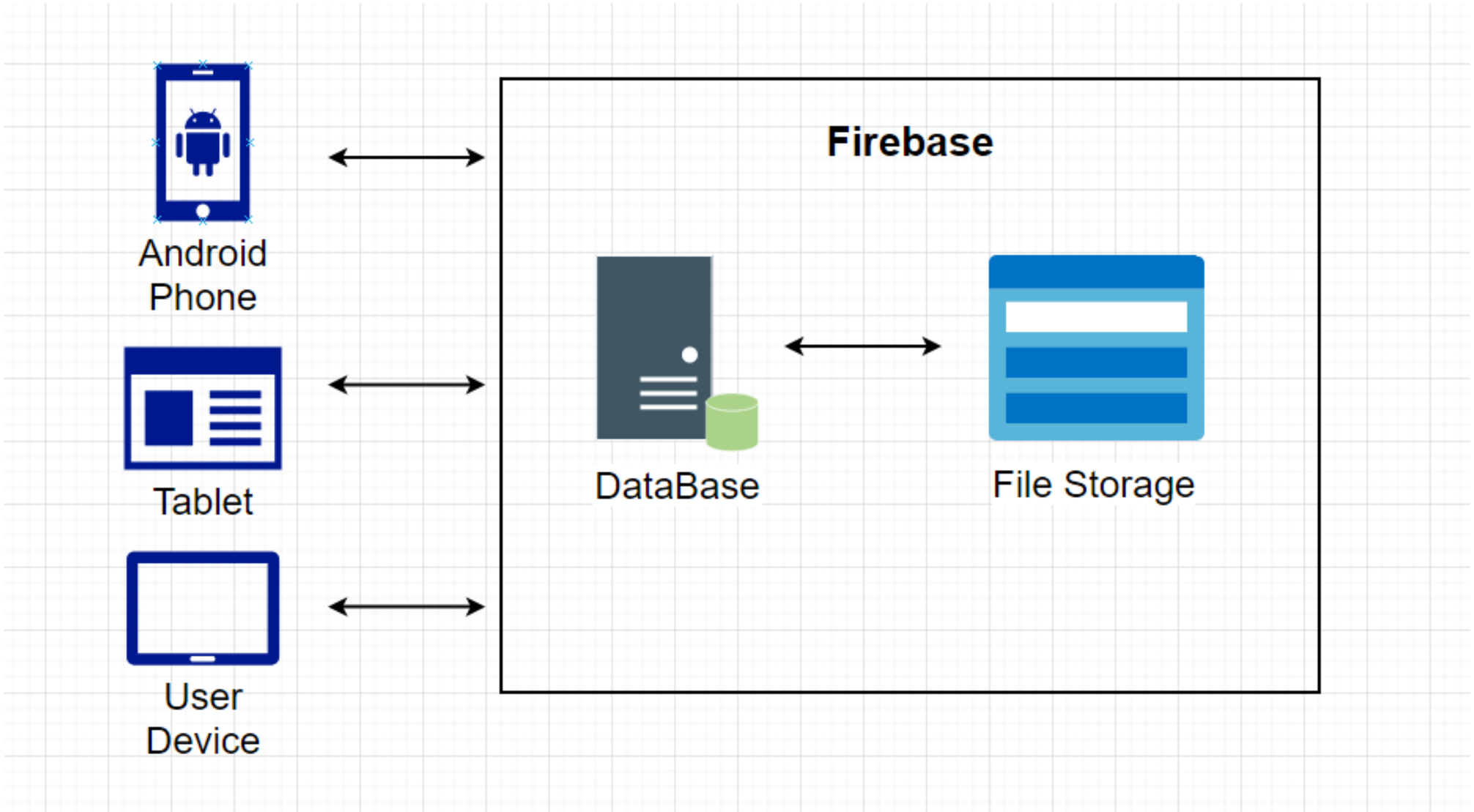


					ДП 6330.03.000 СБД
Розробн.					Схема бази даних
Керівн.					
Консульт.					
		Проскура С.Л.			КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63

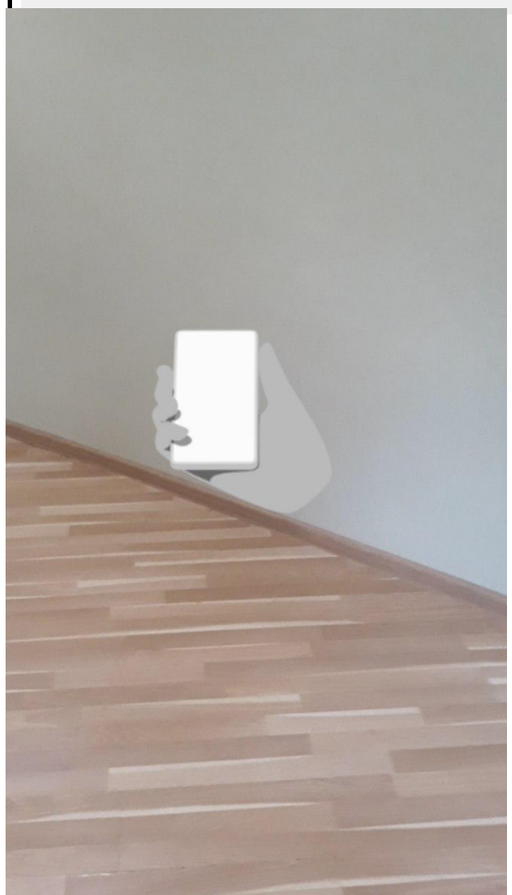

CameraActivity
+ LOGGER: Logger
+ PERMISSIONS_REQUEST: int
+ PERMISSION_CAMERA: String
+ previewWidth: int
+ previewHeight: int
+ handler: Handler
+ handlerThread: HandlerThread
+ onCreate(Bundle savedInstanceState): void
+ getRgbBytes(): int[]
+ getLuminanceStride(): int
+ getLuminance(): byte[]
+ onPreviewFrame(byte[] bytes, Camera camera): void
+ onImageAvailable(ImageReader reader): void
+ onStart(): void
+ onResume(): void
+ onPause(): void
+ onStop(): void
ClassifierActivity
+ LOGGER: Logger
+ DESIRED_PREVIEW_SIZE: Size
+ TEXT_SIZE_DIP: float
+ rgbFrameBitmap: Bitmap
+ lastProcessingTimeMs: long
+ classifier: Classifier
+ borderedText: BorderedText
+ imageSizeX: int
+ imageSizeX: int
+ getLayoutId(): int
+ getDesiredPreviewFrameSize(): Size
+ onPreviewSizeChosen(Size size, int rotation): void
+ processImage(): void
+ onInferenceConfigurationChanged(): void
+ recreateClassifier(Model model, Device device, int numThreads): void
BorderedText
+ interiorPaint: Paint
+ exteriorPaint: Paint
+ textSize: float
+ BorderedText(float textSize)
+ BorderedText(int interiorColor, int exteriorColor, float textSize)
+ setTypeface(Typeface typeface): void
+ drawText(Canvas canvas, float posX, float posY, String text): void
+ drawLines(Canvas canvas, float posX, float posY, Vector<String> lines): void
+ setInteriorColor(int color): void
+ setExteriorColor(int color): void
+ getTextSize(): float
+ setAlpha(int alpha): void
+ getTextBounds(String line, int index, int count, Rect lineBounds): void
+ setTextAlign(Align align): void
ClassifierFloatEfficientNet
+ IMAGE_MEAN: float
+ IMAGE_STD: float
+ PROBABILITY_MEAN: float
+ PROBABILITY_STD: float
+ ClassifierFloatEfficientNet(Activity activity, Device device, int numThreads)
+ getModelPath(): String
+ getLabelPath(): String
+ getPreprocessNormalizeOp(): TensorOperator
+ getPostprocessNormalizeOp(): TensorOperator
CameraConnectionFragment
+ LOGGER: Logger
+ MINIMUM_PREVIEW_SIZE: int
+ ORIENTATIONS: SparseIntArray
+ FRAGMENT_DIALOG: String
+ CameraConnectionFragment(ConnectionCallback connectionCallback, OnImageAvailableListener imageListener, int layout, Size inputSize)
+ chooseOptimalSize(Size[] choices, int width, int height): Size
+ newInstance(ConnectionCallback callback, OnImageAvailableListener imageListener, int layout, Size inputSize): CameraConnectionFragment
+ showToast(String text): void
+ onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): View
Logger
+ DEFAULT_TAG: String
+ DEFAULT_MIN_LOG_LEVEL: int
+ IGNORED_CLASS_NAMES: Set<String>
+ tag: String
+ messagePrefix: String
+ minLogLevel: int
+ Logger(Class<?> clazz)
+ Logger(String messagePrefix)
+ Logger(String tag, String messagePrefix)
+ Logger()
+ Logger(int minLogLevel)
+ getCallerSimpleName(): String
+ setMinLogLevel(int minLogLevel): void
+ isLoggable(int logLevel): boolean
+ toMessage(String format, Object args): String
Classifier
+ LOGGER: Logger
+ Model: enum
+ Device: enum
+ MAX_RESULTS: int
+ tfliteModel: MappedByteBuffer
+ gpuDelegate: GpuDelegate
+ imageSizeX: int
+ imageSizeY: int
+ nnApiDelegate: NnApiDelegate
+ tflite: Interpreter
+ labels: List<String>
+ inputImageBuffer: TensorImage
+ outputProbabilityBuffer: TensorBuffer
+ create(Activity activity, Model model, Device device, int numThreads): Classifier
ClassifierFloatMobileNet
+ IMAGE_MEAN: float
+ IMAGE_STD: float
+ PROBABILITY_MEAN: float
+ PROBABILITY_STD: float
+ ClassifierFloatMobileNet(Activity activity, Device device, int numThreads)
+ getModelPath(): String
+ getLabelPath(): String
+ getPreprocessNormalizeOp(): TensorOperator
+ getPostprocessNormalizeOp(): TensorOperator

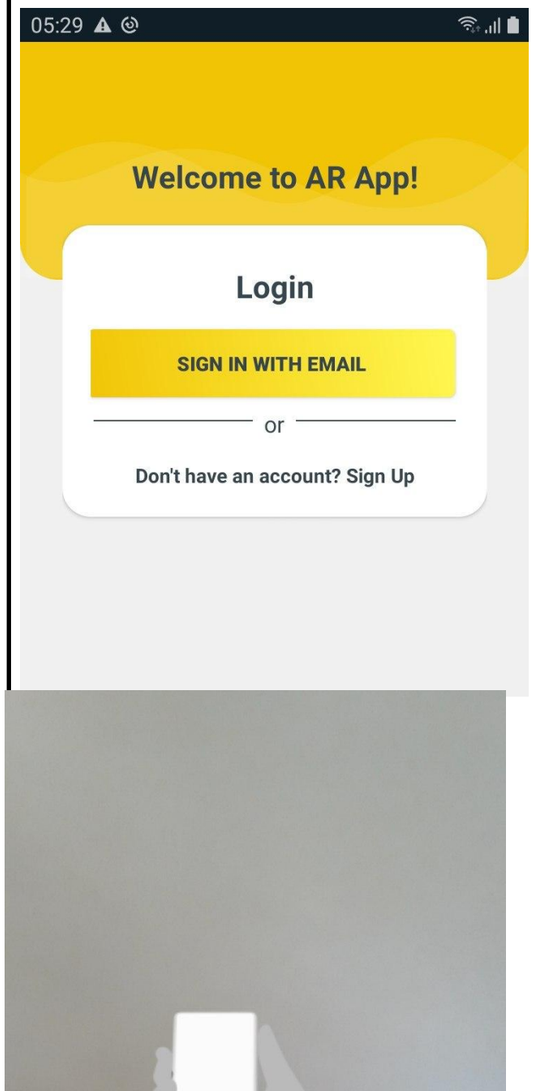


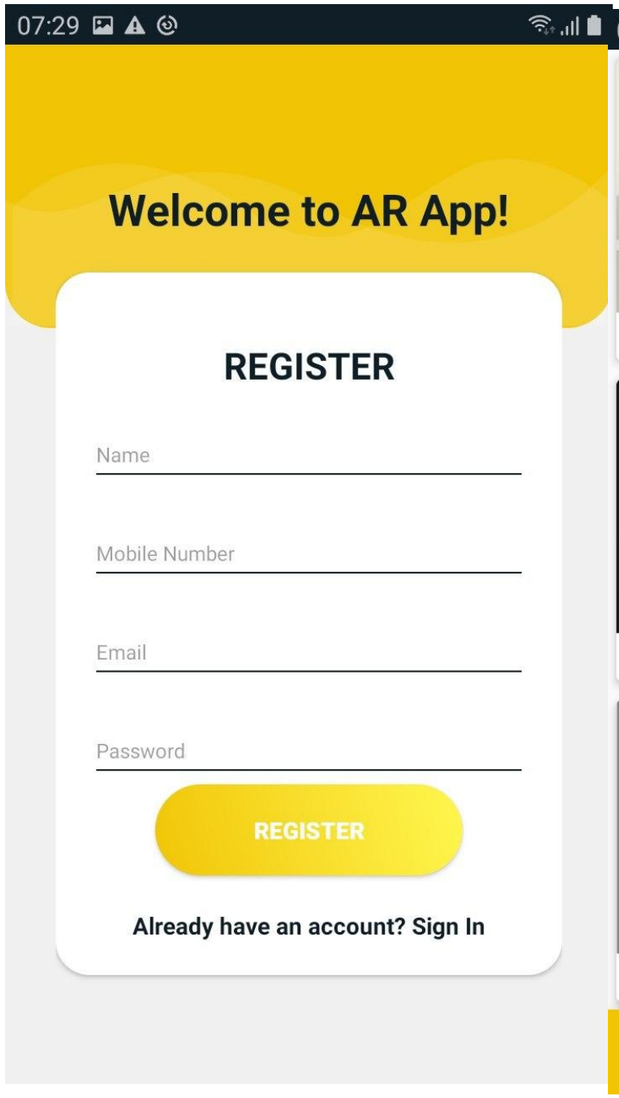
					ДП 6330.05.000 ССП	
Розробн.					Схема структурна послідовності	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63
Керівн.						
Консульт.						
		Проскура С.Л.				

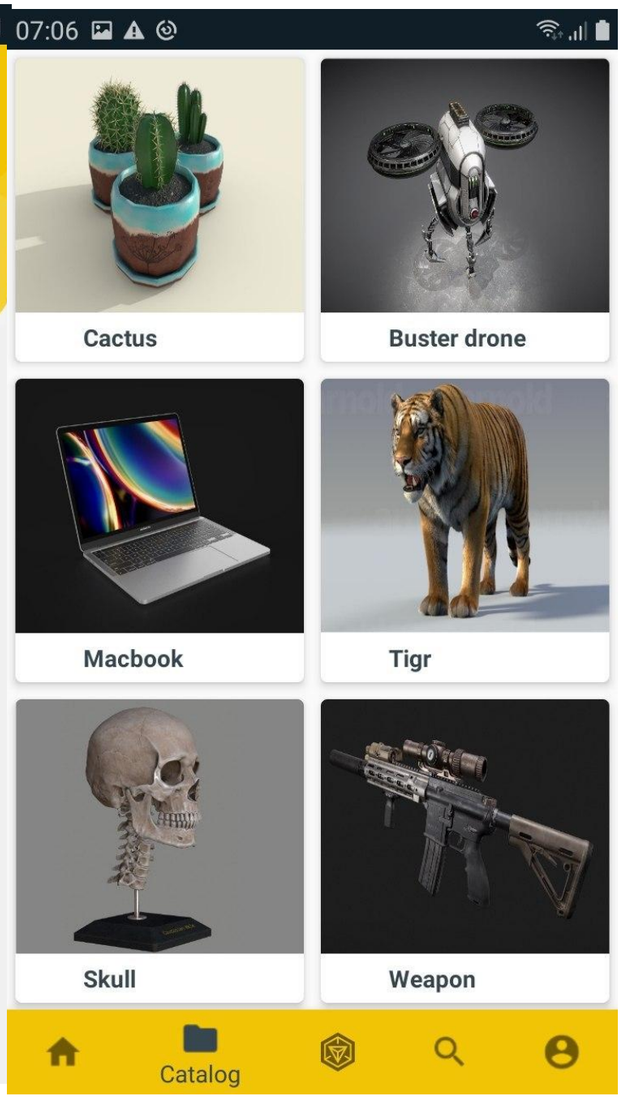


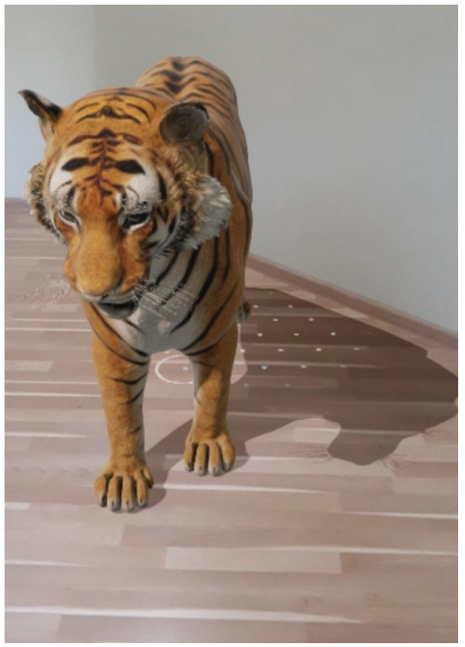
					ДП 6330.06.000 АПЗ	
Розробн.					Архітектура програмного забезпечення	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63
Керівн.						
Консульт.						
		Проскура С.Л.				











					ДП 6330.07.000 ЕК	
Розробн.				Креслення вигляду екранних форм		
Керівн.						
Консульт.						
	Проскура С.Л.				КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63	